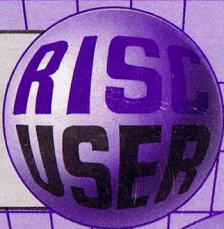# RISC USER

A

RISC
USER

Image Wrap and Spin

THE MAGAZINE AND SUPPORT GROUP
EXCLUSIVELY FOR USERS OF THE ARCHIMEDES

RISC USER

CONTENTS

EDITORIAL

As the launch of RISC OS draws nearer, so more information becomes available on exactly what users will get in their upgrade kit. One of this month's articles makes an attempt to pin down the major software enhancements in RISC OS, and to give a shortlist of the applications software which will accompany the ROMs - though shortage of space on the new Welcome discs may mean that some items, such as the additional fonts, may not after all be included.

As far as documentation is concerned, Acorn plan to include two manuals with the upgrade: a new Welcome Guide, and a RISC OS User Guide. This latter will detail the Archimedes' major star commands, including of course those new with RISC OS, and will provide an introduction to multi-tasking and the Desktop. Neither manual will cover the Basic language, which remains virtually unaltered. Such new features as have been introduced will be covered in subsequent editions of the Basic User Guide for the machine.

Acorn are also working on a new version of the Programmer's Reference Manual, which will probably run to three volumes, and will not be available for some months. Neither this nor the Basic User Guide will be offered as an upgrade, though the whole of the present Basic User Guide is totally applicable to RISC OS, and virtually everything in the present Programmer's Manual similarly applies.

This underlines the nature of the RISC OS upgrade, which provides features *in addition* to those currently available under Arthur 1.2. In fact, even though RISC OS may be installed in your machine, you can virtually ignore it, and use the machine as if it still contained the old 1.2 OS, with almost all software running as it did before. Once you make use of the new features however, you have something akin to a completely new machine, with a set of extremely powerful and impressive features.

The only downside to the new operating system is that it has become significantly more complex from a programming point of view, and this particularly applies to the whole WIMP system, which is now central to the machine's operation in any multi-tasking application. But this is where RISC User comes in. In the coming months we will try to throw some light on the more important features of RISC OS, though without of course compromising our coverage of other aspects of this first class machine.

*This month's telesoftware password is **strawberry**.*
(see BEEBUG pages on Micronet)

NEW COMPUTER?

Acorn was quick to react to the story about the new Archimedes 205 that was carried in last month's RISC User. Acorn confirmed that there will be a new machine, although it will *not* be a cut-down version of the 310. Indeed, one source has said that the new machine, codenamed the "XXXX", will not be marketed under the Archimedes name. Acorn neither confirmed nor denied the price of £599 that we suggested, although they did confirm that the machine would be launched in the summer.

It is also believed that Acorn is planning some changes to the existing Archimedes series. These will most probably be announced to coincide with the RISC OS release in April or May.

ANIMATE YOUR IMAGES

Film-Maker from Silicon Vision is an animation system for use with the company's *SolidCAD* and *Realtime Solids Modeller* systems. It allows individual frames to be linked into an animated sequence, and various effects applied to the images. For example, the camera angle can be altered to produce a variety of effects. *Film-Maker* costs £79.95 inc. VAT. Further details can be obtained from Silicon Vision, Signal House, Lyon Road, Harrow HA1 2AG, tel. 01-861 2173.

Not to be deterred, Ace Computing who produce *Euclid*, a rival to Silicon Vision's *SolidCAD*, has released its own animator, called *Mogul*. This offers similar features to *Film-Maker*, and costs £20 inc. VAT to registered Euclid users. Further details from Ace Computing, 27 Victoria Road, Cambridge CB4 3BW, tel. (0223) 322559.

ARC FAX

Computer Concepts has said that its Archimedes Fax podule will be available in the near future. The system has now passed the main hurdles involved in obtaining BABT approval, and it only remains for the approval to be officially registered. Computer Concepts has also finally announced the price for the Fax system. The actual Fax podule, which is little more than a fast modem with error correction and all the necessary software, will cost £573.85 inc. VAT. However, this will only enable you to send on-screen images. To enable written documents to be sent, you will also need a scanner, and this pushes the price up to £918.85. More details can be obtained from Computer Concepts, Gaddesden Place, Hemel Hempstead, Herts HP2 6EX, tel. (0442) 63933.

ETHERNET FOR THE ARCHIMEDES

Acorn has announced an Ethernet interface podule for the Archimedes. This will allow machines to be networked using the very high speed local area network developed by Rank Xerox. It will also mean that companies and establishments that already have an Ethernet system can use the Archimedes as a low-cost intelligent terminal.

The Ethernet podule will come in two forms, one for standard Ethernet systems, and the other for the lowercost thin cable Ethernet. The actual hardware is the same as used in the new Acorn Unix machine, but because of software development time, the Archimedes Ethernet interface will not be available until September, and no price has been announced yet.

MORE ART

Atelier, from Minerva, is the latest Archimedes art package to be released. The program works in 256 colours, and offers all the features that people have come to expect of Archimedes art packages, including the ability to distort an image into a particular shape. Also included is an independent sprite designer that can be used to define sprites for use in your own programs. *Atelier* costs £99.95 inc. VAT, and more details can be obtained from Minerva Software, 69 Sidwell Street, Exeter, EX4 6PH, tel. (0392) 437756.

CONGRATULATIONS

At a recent meeting of Acorn's *Premier Dealer League*, consisting of the top 25 Acorn dealers, BEEBUG was one of three companies which were presented with awards for their support of the Archimedes range. The other awards went to Watford Electronics and Twyford based MAV Ltd. **RU**

What's new in RISC OS

A checklist compiled by Lee Calcraft.

THE DESKTOP

The Archimedes Desktop has been vastly improved, and supports multi-tasking through the WIMP. New tasks are started by double-clicking on an application directory from within the Desktop. The Desktop now works in all modes, automatically scaling sprites and text characters where appropriate.

Ultimate control over multi-tasking is provided by the Task Manager, which may be called from the Desktop to alter configuration settings, and to set the RAM available for the next task.

The revamped Desktop is now a real module, and does not overwrite user workspace. It is much more powerful, and you may perform many more operations directly from it. These include: delete, rename, copy or change access of a file or directory, format or backup a disc; and most important of all, run a file or application. Copying is performed by dragging a file to the new directory.

To run an application, just double-click on the application directory, or on an associated data file. Alternatively, you can drag a file onto the application's icon, and this will start the application, and load the file into it. If you need to open an application directory, you must double click on it while pressing the Shift key.

The WIMP. A number of new WIMP calls have been introduced to cater for multi-tasking. The WIMP now allows window dragging with continual refresh of the dragged window. To implement this, use *Configure WimpFlags 15 (0 resets it).

BASIC

The new version of Basic supplied with RISC OS has a number of enhancements. These are mainly concerned with array operations and the assembler. There is also a number of additional routines exported by the CALL statement.

FILING SYSTEM

ScreenSave/Load. *ScreenSave and *ScreenLoad have been considerably speeded up when used with a full-sized graphics window.

Obey Files. A new file of type &FEB has been introduced. This is similar to an Exec file except that the contents of the file are sent directly to the operating system. Obey files are not closed when an application is started, and will continue to be obeyed once the application ends. The contents of Obey files are not normally echoed to the screen.

Copy T. A new *COPY option, using the parameter "T", will cause a directory structure, but not its component files, to be copied.

RAM Filing System. A complete RAM filing system has been implemented allowing fast saving and loading to an area of designated RAM (use *Configure RamFsSize).

HourGlass. A dynamic hourglass pointer can be made to appear during filing and other operations to indicate the time during which the processor is busy. The hourglass appears automatically during network filing operations.

DISC

The E Format. A new 800K disc format is provided which fragments files, and so does not need repeated compacting (though *Compact can still be used to de-fragment files). The new format also holds a back-up copy of the space allocation map, so reducing the likelihood of disc errors. The new command *CheckDisc checks this data.

Disc Names. On RISC OS discs are referred to by disc name (drive numbers are generally not used). So all discs should be given a meaningful name using *NameDisc.

SPRITES

Sprite Extend Module. This new module adds many sprite SWI calls. Sprites may be scaled (magnified or reduced), and their palette

RISC OS

controlled. VDU commands can be directed to sprites, so that a sprite may be written to even when not on screen, and sprites may be easily converted for use as pointers.

VARIOUS

New Modes. Screen modes greater than 20 have been altered and vastly extended. Mode 21 now gives 640x512 resolution in 256 colours (using 320K, and needing a multi-sync). Mode 24 gives 1056x256 resolution in 256 colours, and will work with an ordinary monitor. Modes 25-28 cater for IBM compatible VGA monitors.

International Keyboard. This new module allows the keyboard to be reconfigured to accommodate the currently selected country. An unwanted side-effect is that the Alt key can no longer be used like Shift or Ctrl. To regain this facility, use *UnPlug InternationalKeyboard.

*Help. This has been extended so that *Help <module name> will list the star commands and configuration options provided by that module.

BUNDLED APPLICATIONS

A collection of disc-based applications is supplied with RISC OS. The majority of these will multi-task from the Desktop.

MAJOR APPLICATIONS

ArcEdit. This is a full-feature text editor, which can use normal or fancy fonts. It has cut and paste facilities, and can operate on multiple windows. It is ideal for creating Exec and Obey files, but is equally at home with ordinary text, or with binary files.

ArcDraw. This is a powerful object-orientated drawing package with many of the features of CAD software. It can handle splines, and objects may be rescaled and rotated. Text in fancy fonts can be placed anywhere within the active window, and text from ArcEdit can be imported.

ArcPaint. By contrast this is a pixel-orientated painting program, and replaces that supplied on the original Welcome disc. Again it multi-tasks through the WIMP, and any number

of windows may be opened. It is ideal for producing and editing sprites for use as icons etc.

MINOR APPLICATIONS

Maestro. This is a new version of the music editor supplied on the original Welcome disc, though it is much less colourful, because it must conform to the multi-tasking palette.

Configure. This is a simple WIMP-based set-up screen for the main configure options such as beep volume, keyboard auto-repeat, and so on.

Magnifier. This shows off the use of the scaling routines in the sprite extend module, and displays a mouse-driven magnifying glass, which will magnify any area of the screen.

Puzzle and Patience. Two multi-tasking games to think about while you format discs. Unfortunately none of the Arc's filing operations are multi-tasked, so everything stops for loads, saves, formats and the like.

TinyDirs. This useful application allows icons of frequently used applications to be posted on the icon bar for instant access.

The inevitable Clock, Alarm and Calculator are also included, as is a small application giving a graphical representation of CPU usage, an electronic mail handler, and the Lander program from the Arthur Welcome disc - though the latter will unfortunately not multi-task.
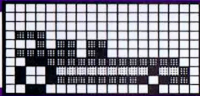
ADDITIONAL ITEMS

Basic Edit. This is now supplied on disc, due to lack of room in the RISC OS ROM. The same is true of the HardCopy module.

65Host and 65Tube. New and improved versions of the 6502 emulator to emulate the model B or a 6502 second processor.

Fonts. Three fonts are supplied on disc:
Corpus Medium
Porterhouse Standard and
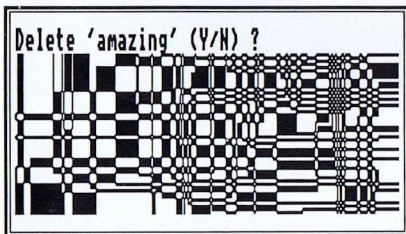*Trinity Medium* RU

A SIMPLE SPRITE MANAGER

Manage your store of resident system sprites with this short piece of code by Mike Ironmonger.

Keeping track of system sprites, and selectively deleting those no longer required is a tiresome task, since *SDelete requires the exact name of any sprite to be deleted. The utility listed here provides a single star command:

*SWipe

which displays each sprite in turn, and asks whether it should be deleted.

The program is extremely short, and is given the file type &FFC. This means that it is a so-called *Transient Utility*, and does not need to be specifically loaded before use. All you need to do is issue the command *SWipe, and the code will be loaded from disc into the RMA area (though it is not a Module as such). It will then execute, and will be automatically cleared after use.

To make use of the program, type it in carefully, save away the source code, and run it - making sure that your disc has a directory named $.Library. The Transient Utility will be assembled and saved to $.Library ready for use.

Delete 'amazing' (Y/N) ?

```
10 REM              >SpriteMan
20 REM Program      Sprite Manager
30 REM Version      A 1.30
40 REM Author       Mike Ironmonger
50 REM RISC User    March 1989
60 REM Program      Subject to Copyright
70 :
80 MODE0:DIM code% 500
90 FORA%=0TO1:P%=code%
100 :
110 [OPT A%*2
120 MOV    R0,#8:SWI   "XOS_SpriteOp"
130 MOVVS  PC,R14:MOVS  R8,R3
140 BNE    first_sprite
150 :
160 SWI    "XOS_WriteS"
170 EQUS   "No Sprites defined"+CHR$13+
CHR$10+CHR$0
180 MOV    PC,R14
190 :
200 .first_sprite
210 MOV    R0,#135:SWI   "XOS_Byte"
220 MOV    R9,R2:MOV   R7,#1
230 :
240 .next_sprite
250 MOV    R0,#13:MOV   R2,R12
260 MOV    R3,#13:MOV   R4,R7
270 SWI    "XOS_SpriteOp"
280 :
290 MOV    R0,#40:SWI   "XOS_SpriteOp"
300 TEQ    R6,R9:SWIEQ &20100+12
310 SWINE  &20100+22:MOVNE R0,R6
320 SWINE  "XOS_WriteC"
330 MOVNE  R9,R6
340 :
350 SWI    "XOS_WriteS"
360 EQUS   "Delete '"+CHR$0
370 MOV    R0,R12:SWI   "XOS_Write0"
380 SWI    "XOS_WriteS"
390 EQUS   "' (Y/N) ? "+CHR$0
400 :
410 MOV    R0,#34:MOV   R3,#0
420 MOV    R4,R4,LSL #2
430 RSB    R4,R4,#992:MOV   R5,#8
440 SWI    "XOS_SpriteOp"
450 :
460 SWI    "XOS_ReadC":BCS   end
470 TEQ    R0,#ASC"Y":TEQNE R0,#ASC"y"
480 ADDNE  R7,R7,#1:MOVEQ R0,#25
490 SWIEQ  "XOS_SpriteOp"
500 :
510 SUBS   R8,R8,#1:BNE   next_sprite
520 .end
530 MOV    R0,#126:SWI   "XOS_Byte"
540 SWI    &20100+12:MOV   PC,R14
550 ]:NEXT
560 :
570 SYS "OS_File",10,"$.Library.SWipe"
,&FFC,,code%,P%
580 PRINT "'*SWipe' command now instal
led in library"
```
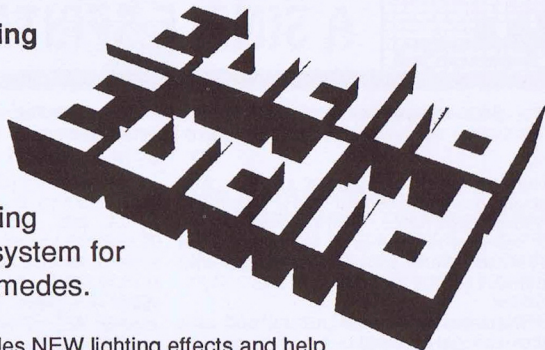
RU

IMAGE WRAP AND SPIN

Use this short program by Nathan Williams to wrap flat images around a sphere, and spin them anywhere on the screen.

> **You will need at least 50K of sprite space to run this program.**

You may have seen Arc screen displays of rotating worlds similar to the BBC1 world logo. The technique used to generate these is to produce a single flat world map (i.e. a Mercator projection), and then wrap it around a sphere. To give the impression of rotation, a series of wrapped images are created from the original flat map, each differing from the next by a few degrees. When the sequence is played back a very powerful impression of rotation is created.
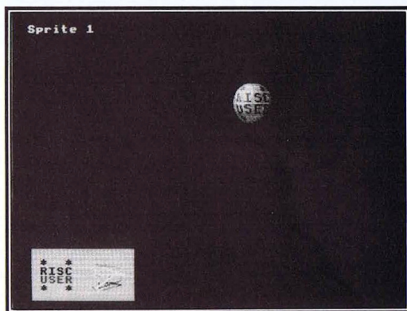
Image wrapping in progress

The accompanying program uses similar techniques, but since it is written in Basic, and has been made modular, it is easy to adjust to provide different effects. If you type it in and run it as it stands, you will see a rectangular image containing the words "RISC USER" at the bottom left hand corner of the screen. This serves as the image to be wrapped. The wrapping process begins immediately, and you will see a spherical transform of the flat map slowly build up in the upper right-hand quadrant of the screen.

As soon as the image is complete, it will be stored in the machine's sprite area as a sprite

named *Sphere1,* and a new spherical image, similar to the first, but offset by a small rotational angle, will be generated. This will be stored as the sprite *Sphere2.* The process will continue until the sphere has been mapped from all angles, and the display will then begin. This simultaneously rotates three spheres, each carrying the RISC USER title.

It is possible to save the sprites used by the program. To do this, press Escape during the display phase, and use:
```
*SSAVE mysprites
```
To make use of the sprites at any time, use:
```
*SLOAD mysprites
```
Then, with the program in memory, type:
```
PROCplay
```

As I mentioned earlier, the program is easily customisable. You can change the original image, the size of the sphere, the angle of rotation, and the lighting and reflective properties of the sphere. You can also alter the program so that it generates just a single wrapped image rather than a whole sequence.

Firstly, the original image: this is generated by PROCscreen, which may be replaced by your own routine. The image used must be twice as wide as it is high, and its height must be equal to the value held in the variable map% (in PROCinit). This is currently set to 96 for an image 192x96 graphics units in size.

The size of the resultant sphere is determined by the variable rad%, set up in PROCinit. This is currently set to 75. Remember that increasing this will radically affect the space needed to store the resultant sprites, and the time taken to generate them.

To alter the rotation from vertical to horizontal, change the main FOR loop variable in line 120 from *anglev* to *angleh.*

The angle of lighting of the sphere is determined by the x, y and z co-ordinates of the

light source, held in the variables lx, ly and lz (defined in PROCinit), which range in value from -1 to +1. To remove the shading effect, set the variable *shaded* to FALSE in line 260, and to remove the highlighting effect, set the variable *highl* to FALSE in line 270.
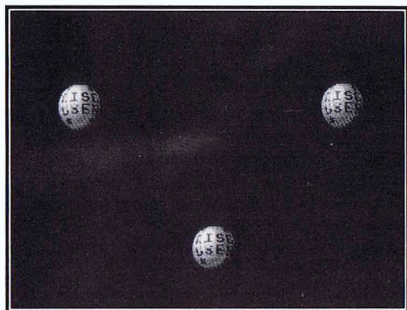
A sequence of sprites created by the program

If the shading effect is to work fully, then the original flat image should only use colours of maximum brightness. That is to say for example, if there is a blue component in a given colour, it must be maximum blue. The reason for this is that shading is achieved by reducing the intensity of each component of a given colour, and if colour components are at half intensity at the start, the result of shading may be to alter the colour rather than the shade of a given pixel. If the colours of an image do not contain full intensity components, such as would be the case for a digitised image, for example, it will be best to remove the shading option; though you should be aware that the current program does not pick up tint information when transferring each pixel to the sphere.

There is already plenty to experiment with here, but there is even greater scope for altering PROCplay. At present, this displays three spinning spheres. But you could alter the plotting positions so that one or more spheres

appears to bounce around the screen. Remember, though, that if you intend to have more than one moving sphere on screen, then you will need to use bank switching to achieve smooth animation, since you will not be able to use Exclusive OR plotting to erase previous images.

```
  10 REM            >Wrap
  20 REM Program    Image Wrap
  30 REM Version    A 0.3
  40 REM Author     Nathan Williams
  50 REM RISC User March 1989
  60 REM Program    Subject to Copyright
  70 :
  80 ON ERROR MODE 13:PRINT REPORT$;" @
line ";ERL:END
  90 MODE13:OFF
 100 PROCscreen
 110 PROCinit
 120 FOR anglev=0 TO 2*PI STEP PI/15
 130 PROCsphere(rad%,anglev,angleh)
 140 PROCgetsphere(rad%)
 150 NEXT
 160 PROCplay
 170 END
 180 :
 190 DEFPROCinit
 200 DIM col%(20,3)
 210 ORIGIN 840,712
 220 FOR C%=0 TO 13
 230 READ col%(C%,1),col%(C%,2),col%(C%
,3)
 240 NEXT
 250 lx=-0.55:ly=0.5:lz=0.7:REM Light
 260 shaded=TRUE:      REM Shading On
 270 highl=TRUE:       REM Highlight On
 280 map%=96:          REM Image Height
 290 rad%=75:          REM Sphere Radius
 300 HP=PI/2:sprite%=1
 310 anglev=0:angleh=0
 320 ENDPROC
 330 :
 340 DEFPROCscreen
 350 GCOL 0,12:RECTANGLE FILL 0,0,384,1
92
 360 COLOUR 140 TINT 240:COLOUR 0
 370 PRINTTAB(1,28);"RISC"
 380 COLOUR 3
```

```
 390 PRINTTAB(1,29);"USER"
 400 COLOUR 32:PRINTTAB(0,27);"  *  *"
 410 PRINTTAB(0,30);"  *    *"
 420 MOVE 300,100
 430 FOR J%=1 TO 20
 440 GCOL 0,RND(63)
 450 DRAW 224+RND(128),32+RND(128)
 460 NEXT
 470 ENDPROC
 480 :
 490 DEFPROCsphere(R%,anglev,angleh)
 500 COLOUR 63:COLOUR 128 TINT 0
 510 PRINTTAB(0,1)"Sprite ";sprite%
 520 FOR Y%=R% TO -R% STEP -4
 530 FOR X%=R% TO -R% STEP -4
 540 Z%=FNgetz(X%,Y%)
 550 IF Z%>0 THEN
 560 PROCtransform(X%,Y%,Z%,anglev,angl
eh)
 570 PROCgcol(X,Y)
 580 PLOT 69,X%,Y%
 590 ENDIF
 600 NEXT
 610 NEXT
 620 ENDPROC
 630 :
 640 DEFPROCgetsphere(R%)
 650 MOVE -R%,-R%:MOVE R%,R%
 660 OSCLI("SGET sphere"+STR$(sprite%))
 670 sprite%=sprite%+1
 680 ENDPROC
 690 :
 700 DEFFNgetz(X%,Y%)
 710 Z%=rad%^2-X%^2-Y%^2
 720 =Z%
 730 :
 740 DEFPROCtransform(X%,Y%,Z%,anglev,a
ngleh)
 750 newz=SQR(Z%)*COS(anglev)-Y%*SIN(an
glev)
 760 newy=Y%*COS(anglev)+SQR(Z%)*SIN(an
glev)
 770 theta=X%/SQR(X%^2+newz^2)
 780 X=rad%*ASN(theta)
 790 Y=rad%*ATN(newy/SQR(newz^2+X%^2))
 800 IF newz<0 THEN X=(PI*R%)-X
 810 X+=rad%*angleh
 820 ENDPROC
 830 :
 840 DEFPROCgcol(X,Y)
```

```
 850 colour%=FNcol(X,Y)
 860 brightness=FNbright(X%,Y%,SQR(Z%))
 870 highlight%=(brightness*13+RND(1)/1
5)
 880 light%=(brightness*12+RND(1))
 890 IF shaded THEN colour%=colour% AND
 col%(light%,1):tint%=col%(light%,2) ELS
E tint%=0
 900 IF highl THEN colour%=colour% OR c
ol%(highlight%,3)
 910 GCOL 0,colour% TINT tint%
 920 ENDPROC
 930 :
 940 DEFFNcol(X,Y)
 950 x%=X/rad%/HP*map%:y%=Y/R%/HP*map%
 960 x%=(x%+map%) MOD (map%*4):y%=(y%+m
ap%) MOD (map%*2)
 970 =POINT(x%-840,y%-712)
 980 :
 990 DEFFNbright(x%,y%,z%)
1000 x=x%/rad%:y=y%/rad%:z=z%/rad%
1010 =(x*lx+y*ly+z*lz)/2+.5
1020 :
1030 DATA %01010101,63,0,%01010101,63,0
1040 DATA %01010101,127,0,%01010101,191
,0
1050 DATA %10101010,63,0,%10101010,127,
0
1060 DATA %10101010,191,0,%10101010,255
,0
1070 DATA %11111111,63,0,%11111111,127,
0
1080 DATA %11111111,191,0
1090 DATA %11111111,255,0
1100 DATA %11111111,255,%010101
1110 DATA %11111111,255,%101010
1120 :
1130 DEFPROCplay
1140 MODE 13
1150 ORIGIN 0,0
1160 REPEAT
1170 FOR J%=1 TO 30
1180 OSCLI("SCHOOSE sphere"+STR$(J%))
1190 PLOT &ED,0,500
1200 PLOT &ED,880,500
1210 PLOT &ED,440,50
1220 WAIT:WAIT
1230 NEXT
1240 UNTIL FALSE
1250 ENDPROC
```

RU

NEW FONT STYLES

This short relocatable module by Peter Arnold provides 72 new styles for the Arc's normal screen font.

The Font Masher relocatable module is designed to bridge the gap between the Archimedes' normal screen font (BFONT and its ISO variants), and the anti-aliased fonts that are loaded from disc. The normal font is quick and simple, but there is only one style. The anti-aliased fonts take time to load, and limit the palette available for other purposes in all but the 256 colour modes. Font Masher provides seventy-two varied fonts that are used in exactly the same way as BFONT, and which retain BFONT's great speed.

To make use of the module, type in the program in listing 1, and save it to disc. When run, it will assemble a module, and automatically save it to disc under the name RMmash. To use the module, just type:

```
*RMmash
```

If there is insufficient room in the RMA, type:

```
*Quit
*RMmash
*Basic
```

Now you can change fonts with the new star command *Mash. It takes a numeric parameter from 0 to 255, which defines the new font. *Mash 0 sets the font to normal. The table shows how the parameter is determined. If you want bold text, use:

```
*Mash 128
```

If you want bold and underlined, use:

```
*Mash 136
```

i.e. just add the codes for the features that you require.

Bit Number	Font Style	Decimal
0	Italics	1
1	Backward Italics	2
2	Feint	4
3	Underlined	8
4	Inverse	16
5	Superscript	32
6	Subscript	64
7	Bold	128

Table 1. Typestyles and corresponding parameters

There are three pairs of font styles that cannot be used together for obvious reasons. They are:

Italics and Backward Italics
Feint and Bold
Superscript and Subscript.

Having tried some of the new font styles, you will probably find a number which you want to use. To make reference to them easier, you can use the Arc's *SET Alias$ command to give them appropriate names. For example, if you issue the following strings:

```
*SET Alias$Bold Mash 128
```

and

```
*SET Alias$Normal Mash 0
```

then every time that you type:

```
*Bold
```

the bold font will be engaged. The command:

```
*Normal
```

will reset the style to normal.

The program in listing 2 gives a short demonstration of some of the styles available.

A display of the output from Listing 2

Listing 1

```
10 REM        >Masher
20 REM Program  New Font Styles
30 REM Version   A 0.02
```

13

```
 40 REM Author    Peter Arnold
 50 REM RISC User March 1989
 60 REM Program   Subject to copyright
 70 :
 80 DIM R% &1000
 90 FOR I=4 TO 7 STEP 3
100 P%=0:O%=R%
110 [OPT I
120 EQUD0:EQUD0:EQUD0:EQUD0
130 EQUD title
140 EQUD help
150 EQUD command
160 .title
170 EQUS "FontMasher"
180 EQUB 0
190 ALIGN
200 .help
210 EQUS "Font Masher"+CHR$9+"1.00 ("+
MID$(TIME$,5,11)+")"
220 EQUB 0
230 ALIGN
240 .command
250 EQUS"Mash"
260 EQUB 0
270 ALIGN
280 EQUD start
290 EQUD &00010001
300 EQUD mashsyntax
310 EQUD mashhelp
320 EQUD 0
330 .mashsyntax
340 EQUS "Syntax: *Mash <0-255>"
350 EQUB 0
360 .mashhelp
370 EQUS"*MASH with a number affects t
he characters shown on the screen. The e
ffects depend on the setting of the lowe
r eight bits of the parameter. Use *MASH
 0 to reset the font."
380 EQUB 0
390 ALIGN
400 :
410 .start
420 STMFD R13!,{R0-R12,R14}
430 MOV R1,R0:MOV R0,#&30000000
440 SWI 33:BVS end
450 MOV R6,R2:MOV R0,#20
460 SWI 6:CMP R6,#0
470 BNE carryon
480 B end
490 :
500 .carryon
510 MOV R3,#32
520 .getshape
530 ADR R1,block:STRB R3,[R1,#0]
540 MOV R0,#10:SWI 7
550 .go
560 AND R7,R6,#1:TEQ R7,#1
570 BNE go2:MOV R8,#1
580 MOV R9,#1:MOV R10,#31
590 .italics1
600 BL rotate:ADD R8,R8,#1
610 CMP R8,#4:BLT italics1
620 MOV R9,R10:MOV R8,#6
630 .italics2
640 BL rotate:ADD R8,R8,#1
650 CMP R8,#9
660 BLT italics2
670 .go2
680 AND R7,R6,#2:TEQ R7,#2
690 BNE go3
700 AND R12,R6,#1:CMP R12,#1
710 BEQ go3:MOV R8,#1
720 MOV R9,#31:MOV R10,#1
730 .Bitalics1
740 BL rotate:ADD R8,R8,#1
750 CMP R8,#4:BLT Bitalics1
760 MOV R9,R10:MOV R8,#6
770 .Bitalics2
780 BL rotate:ADD R8,R8,#1
790 CMP R8,#9:BLT Bitalics2
800 .go3
810 AND R7,R6,#4:TEQ R7,#4
820 BNE go4
830 MOV R8,#1
840 .feint
850 LDRB R7,[R1,R8]
860 MOV R9,R7,ROR #1
870 AND R7,R7,R9
880 STRB R7,[R1,R8]
890 ADD R8,R8,#1:CMP R8,#9
900 BLT feint
910 .go4
920 AND R7,R6,#8:TEQ R7,#8
930 BNE go5
940 .underline
950 LDRB R7,[R1,#8]
960 ORR R7,R7,#170
970 STRB R7,[R1,#8]
980 .go5
```

```
 990 AND R7,R6,#16:TEQ R7,#16
1000 BNE go6:MOV R8,#1
1010 .inverse
1020 LDRB R7,[R1,R8]
1030 MVN R7,R7
1040 STRB R7,[R1,R8]
1050 ADD R8,R8,#1:CMP R8,#9
1060 BLT inverse
1070 .go6
1080 AND R7,R6,#32:TEQ R7,#32
1090 BNE go7
1100 MOV R8,#1:MOV R9,#1
1110 .superscript
1120 LDRB R10,[R1,R8]
1130 STRB R10,[R1,R9]
1140 ADD R8,R8,#2:ADD R9,R9,#1
1150 CMP R8,#9
1160 BLT superscript
1170 MOV R8,#5:MOV R9,#0
1180 .blankbottom
1190 STRB R9,[R1,R8]
1200 ADD R8,R8,#1:CMP R8,#9
1210 BLT blankbottom
1220 .go7
1230 AND R7,R6,#64:TEQ R7,#64
1240 BNE go8
1250 AND R12,R6,#32:CMP R12,#32
1260 BEQ go8:MOV R8,#7
1270 MOV R9,#8
1280 .subscript
1290 LDRB R10,[R1,R8]
1300 STRB R10,[R1,R9]
1310 SUB R8,R8,#2:SUB R9,R9,#1
1320 CMP R8,#1:BGE subscript
1330 MOV R8,#1:MOV R9,#0
1340 .blanktop
1350 STRB R9,[R1,R8]
1360 ADD R8,R8,#1:CMP R8,#5
1370 BLT blanktop
1380 .go8
1390 AND R7,R6,#128:TEQ R7,#128
1400 BNE define
1410 AND R12,R6,#4:CMP R12,#4
1420 BEQ define:MOV R8,#1
1430 .bold
1440 LDRB R7,[R1,R8]
1450 MOV R9,R7,ROR #1:ORR R7,R7,R9
1460 STRB R7,[R1,R8]:ADD R8,R8,#1
1470 CMP R8,#9:BLT bold
1480 .define
1490 MOV R8,#0:MOV R0,#23
1500 ADR R1,block:SWI 0
1510 .vdu23
1520 LDRB R0,[R1,R8]
1530 SWI 0
1540 ADD R8,R8,#1:CMP R8,#9
1550 BNE vdu23
1560 ADD R3,R3,#1:CMP R3,#126
1570 BLS getshape
1580 .end
1590 LDMFD R13!,{R0-R12,R14}
1600 MOV PC,R14
1610 .rotate
1620 LDRB R7,[R1,R8]
1630 MOV R7,R7,ROR R9
1640 STRB R7,[R1,R8]
1650 MOV PC,R14
1660 .block
1670 EQUD 0:EQUD 0:EQUB 0
1680 ]:NEXT
1690 OSCLI"SAVE RMmash "+STR$~R%+"+"+ST
R$~P%
1700 OSCLI"SETTYPE RMmash FFA"
```

Listing 2

```
  10 REM >Mash2Demo
  20 REM Font Style Demonstration
  30 :
  40 *RMMASH
  50 ON ERROR REPORT:IF ERR=17 THEN OSC
LI("MASH 0"):END ELSE PRINT" at line ";E
RL:END
  60 MODE 0
  70 *MASH 0
  80 PRINT'"NORMAL        ABC xyz 123"
  90 *MASH 128
 100 PRINT'"BOLD          ABC xyz 123"
 110 *MASH 4
 120 PRINT'"FEINT         ABC xyz 123"
 130 *MASH 1
 140 PRINT'"ITALIC        ABC xyz 123"
 150 *MASH 8
 160 PRINT'"UNDERLINED    ABC xyz 123"
 170 *MASH 19
 180 PRINT'"INVERSE ITALIC ABC xyz 123"
 190 *MASH 32
 200 PRINT'"SUPERSCRIPT   ABC xyz 123"
 210 *MASH 0
```

RU

**Mike Williams offers some advice on the use of Logistix,
the first and maybe one of the best spreadsheets for the Archimedes.**

I have been using Logistix regularly ever since I first reviewed it for RISC User (Volume 1 Issue 3). Although I have had the opportunity to try other spreadsheets as they have appeared (Matrix-3, GammaSheet and PipeDream for example), I have so far found no real incentive to change. The purpose of this article is to look at some of the more frequently used aspects of Logistix, and to present the fruits of my experiences so far.

Logistix offers four different applications in a single package: spreadsheet, database, presentational graphics and time management. I have to say at the outset, that apart from entering a few worked examples, I have personally found no use as yet for the last. I have used the database facilities, but I find that a separate database system better satisfies my needs. Thus this report will concentrate on the spreadsheet and graphics capabilities, which are probably the most important and most commonly used parts of the package anyway.

It seems to me that there are three key features when using a spreadsheet program. How easy is it to create a spreadsheet in the first place? How easy is it to update the data in your spreadsheet once entered? And, finally, and this applies right from the start, how easy is it to change the structure of your spreadsheet? Logistix works well on all counts, and it is the convenience of its user interface which I feel is largely responsible for this.

KEYBOARD INPUT

Logistix may seem rather tame compared with some of the WIMP-based applications around. No mouse to click away with here. All user input is via the keyboard. The cursor keys provide basic movement around the screen, but Page-Up, Page-Down, F9 and F10 to move a whole screenful at a time (up, down, left, right) are also very useful. My main application has grown as time has passed, and I have a number of sub-spreadsheets located in different areas of the available worksheet.

I've also set up graphics displays for most of these sub-sheets as well, again using different pages. It helps here to plan ahead and be consistent (for example, always keep your graphics programs in the first free page to the right of any spreadsheet). Even when you do, you still sometimes find you have located a blank area of the sheet, and the use of the Home key to return to

the top left-hand corner of the sheet is a convenient way of restoring normality.

Data, i.e. numbers or text, is just entered straight in, being placed by Logistix in the current cursor position. Although you can use Return to terminate data entry, it is usually more convenient to use a cursor key, which not only fixes that piece of data, but allows you to move on automatically to the next input position. However, if you edit a field (/E) you *must* terminate the edit process with Return. You also have to get used to the IBM-style back-arrow key for deletion, and not the more usual BBC Delete key (which seems to have the same effect as Escape here).

> **HINT**
> I find with my own application a not infrequent need to update a number by adding a further value to it. Perhaps because I'm lazy, I use the edit facility to replace the original number in a cell by an expression which is the original number plus (or minus) the new number. I then use the replicate command (/R) with the values option (V) to replace the expression in the cell by the value of the expression. The key sequence is:
> /RV,<Return>

Logistix is generally quite good at deciding what category of input it is being presented with. Text, numbers and expressions can often just be typed in. But other classes of input, such as graphics commands, need to be specified as such (a preceding comma for a graphics command for example). When you are trying to find why some aspect of your sheet is not working, do check that every entry is of the right type.

Logistix helps, because as you move the cursor about the screen, a display line at its foot not only shows what is really in each cell (a formula for example, when otherwise all you would see is the value which results from that formula), but Logistix also tells you into which category the cell contents fall. Enter the PIA command to specify a pie chart, and the command will look the same in the spreadsheet whether the three letters are entered as text or as a graphics command. The difference can be the difference between success and failure, so don't get too complacent about input.

REPLICATION

One of the most useful features of any spreadsheet has to be its ability to replicate (copy)

values or formulae across cells, rows, columns or complete blocks of cells. I have not found Logistix wanting in this respect, though I have found difficulty in always getting things to work as I would wish.

Replicating a column

A major convenience of Logistix is that you seldom if ever need to enter cell references from the keyboard. Moving the cursor around the screen picks these up automatically. Thus if you want to replicate the contents of one column to a neighbouring column, move to the top of the column to be replicated (its cell reference appears automatically on the input line at the foot of the screen), and press:

/RR:<foot>,<start>Return

where <foot> means move the cursor to the foot of the column to be replicated, and <start> means move to the first position of the new column.

Note that when replicating a column or a row you only have to specify the first cell of the new location. It is worth practising with the use of the replicate command as you are likely to use it more than most.

Another useful feature of Logistix is that cell references can be specified as absolute (upper case letters) or relative (lower case letters). However, to make use of this feature you must also select the correct setting in Global Options (/G), otherwise, like me initially, you will begin to feel that this feature is a figment of the documentation.

For example, if at the foot of column A you have the entry +SUM(A1:A9) in cell A10 (the '+' specifying an expression), and you replicate the column A1:A10 into column B, the summation will remain quite unaltered (with the correct global option set). But use the expression +SUM(a1:a9) and on replication into column B this is converted to +SUM(b1:b9). Relative cell references throughout are the norm.

Another example of replication uses the *values* option, and converts a formula into its corresponding value. I find this quite useful to fix a value at a certain point in time to form an historical record from then on (see also Hint).

FORMATS

Although Logistix as might be expected provides a good variety of formats, I have found some frustrations. The format command (/F) allows you to specify the width of a column and the format in which numbers, the lifeblood of any spreadsheet, will appear. You can select one or more rows or columns, or a block of cells, and apply any selected format to just that group of cells. However, fewer options are available when dealing with a block of cells than are available with rows and columns. This is unfortunate.
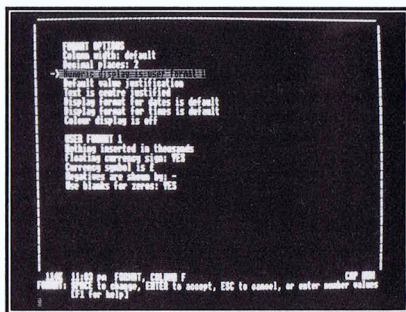
Selecting a format

A format (e.g. number of decimal places) applied to a column affects every single cell on the spreadsheet in that column. Given the paged approach to the use of the spreadsheet, I find I want different formats for different parts of the same column. But formatting a block of cells, which is what you end up doing, does not, for example, permit the number of decimal places to be specified (or fixed), a nuisance when dealing with some figures.

Eight different formats are available for financial data, including the facility for negative amounts to

appear automatically but disconcertingly in red (other, perhaps kinder, formats are also available).

GRAPHICS

I have used the graphics options to display parts of my spreadsheet in either barchart or pie chart format. This does not prove too difficult if you follow the examples given in the manuals. However, I find that most of the displays are quite crude,

Displaying a graph

demonstrating Logistix' regrettable inability to take advantage of the colour range and sophisticated graphics of the Archimedes, while any text captions on a standard colour monitor are well nigh unreadable. Do make sure that the Logistics disc is inserted before displaying any graphs so that any fonts can be correctly accessed. Logistix gets rather unhappy otherwise. All in all, I feel that the graphics are a considerable disappointment.
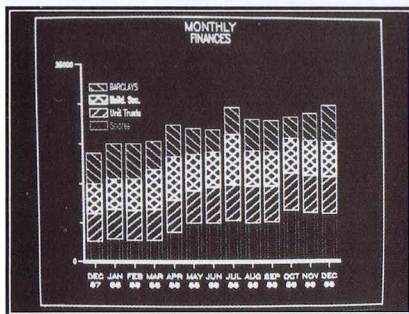
FINAL THOUGHTS

Much of what I have covered has derived, as I said at the start, from my own frequent use of Logistix to maintain what I feel is a reasonably comprehensive spreadsheet application. For much of what I need, Logistix is ideal. However, I certainly do not claim to know everything there is to know on Logistix, and indeed I would welcome any feedback or further information from members, who may be able to add to my own experiences with Logistix for the benefit of others.

Note: Logistix is published by Acorn Computers, and costs £113.85 inc. VAT (available at 5% discount to RISC user members when purchased from BEEBUG). **RU**

Lee Calcraft looks into RISC OS Application Directories and the powerful new Obey files.

In RISC OS the whole Desktop concept has been revamped, and one of the new features to emerge as a result is the *Application Directory*. This is simply a directory in which all resource files for a given application are held, including boot files, sprites, and the application itself. But an application directory is no ordinary directory, it is a runnable object. Running an application directory by double-clicking on its icon will normally start up the application, or in certain cases, install its icon on the icon bar. Fortunately it is a relatively simple matter to set up applications directories.

First of all you need to create a suitable directory. Any name will do providing that its first character is "!". When you have done this, the directory display will register an "A" icon alongside the new name, indicating that it is of type "Application". The new directory will need to contain at least three files:

A file named !Run
A file usually named !RunImage
A sprite file named !Sprites

I will deal with the sprite file first. This should contain one or more icon sprites, and the name of one of these must be exactly the same as that used for the application directory. For example, if the directory name is !QUIZ, the corresponding sprite file (called !Sprites) should contain a sprite called !Quiz (as with all file and sprite names, case is ignored). This sprite is displayed by the Desktop together with the name of the application, whenever the parent directory of the application is opened.

A second sprite may be included for use with the small Desktop icon display. Its name must be the same as that of the main sprite, but preceded by the letters "sp" (e.g. Sp!quiz). Sprites for data files associated with the application may also be included in this file, but this is beyond the scope of the present article.

The easiest way to produce suitable sprites is to use ArcPaint. Icon sprites should normally be a maximum of 68x68 graphics units in size, and should conform to the standard Desktop palette. As a temporary measure, you could *borrow* one of those supplied on the RISC OS Applications discs. Just drag the appropriate sprite file into your new directory. If you need to change its name to match your new directory name, then proceed as follows:

```
*SNEW
*SLOAD !Sprite
*SRENAME !Puzzle !Quiz
*SSAVE !Sprite
```

In the above example we have changed the name of the sprite used for the application !PUZZLE to one suitable for a directory called !QUIZ. Now when you next look at the directory containing your application directory, the new user-defined (or user-borrowed) icon will appear.

OBEY FILES

Now for the !Run file. This will normally be a file of a new type called *Obey* (type &FEB). Obey files resemble Exec files in that they contain executable commands, and may be used to start up an application, just like !Boot files. But they have a number of advantages over Exec files. Firstly, the text which they contain is sent directly to the operating system, and does not rely on the current task sending keyboard input to the O.S. Obey files must therefore contain no Basic instructions, only commands acceptable to the operating system. Secondly the contents of Obey files are not echoed to the screen unless the command *Echo* is issued. And thirdly, Obey files remain in 'obeyance' when an application is running, and then take over once the application ends. This can be useful in different ways. For example, it could be used to make back-ups of data files after the user has terminated an application.

To test this property, use ArcEdit or another text editor to create a text file of type FEB containing the following:

```
Quit
Basic
Echo Why must you leave so soon?
```

If you *RUN this file, Basic will be started up, and you can then run any programs that you wish, and providing that you do not issue a *Close, or Close #0, the Obey file will remain active. At the point that you finally quit Basic, the Obey file will continue, and will display the message before closing. This will work just as well with a single Basic program, providing that it is *RUN. Thus:

```
QUIT
MyProg
Desktop
```

would run the Basic program *MyProg*, and as soon as the program ends (whether naturally, or by pressing Escape), the Desktop will be called.

An application must be able to work properly even if it has been transferred to a new parent directory (as it might be if it had been transferred to a network or to a hard disc). And one important use for the Obey file !Run in any application directory is to record the full path for the given application. The Filer automatically sets the variable Obey$Dir as soon as an Obey file is run, and one of the first tasks of !Run should be to store this information in a suitable environment variable. For example:

```
Set MyProg$Dir <Obey$Dir>
```

This holds the path in the variable MyProg$Dir for subsequent use. One of the final acts in the !Run Obey file is to *Run the file !RunImage. This will usually be the application itself.

A !Run Obey file for an application might therefore contain the following lines:

```
Set MyProg$Dir <Obey$Dir>
Run <MyProg$Dir>.!RunImage %*0
```

Note the way in which the last line uses the convention %*0. If the application is ever run by issuing the command:

```
*<DirectoryName> <parameters>
```

this will ensure that any parameters accompanying this instruction are passed to the application (i.e. passed to !RunImage).

If you examine some of the !Run files on the RISC OS applications discs, you will see that many of them contain additional instructions for setting RunType aliases to ensure that running data files for the application cause the application to start up. Others use the new star commands *RMEnsure and *WimpSlot to check the presence of important modules, and to claim workspace for the application. We will take a closer look at these in a forthcoming issue.

!BOOT

We have not as yet covered the new role of the !Boot file. Any application directory may contain a !Boot file. Again this will typically be an Obey file. This file is automatically run when it is first seen by the Filer. That is to say when a directory is displayed containing the parent directory of the application. Contrast this with the !Run file, which is only run when an application is selected by the user.

If a !Boot file is present it must contain the instruction:

```
*IconSprites !Sprites
```

because if a !Boot file exists, the !Sprite file will not be automatically loaded. One further point: the !Boot file will only be automatically executed providing the icon sprite pool does not contain a sprite whose name suggests that the application is already known to the Filer (i.e. the sprite called !<application name>).

The best way to get to grips with application directories is to experiment with some of your own, and the best source of examples are the directories supplied on the RISC OS Applications discs. In the mean time, a complete application is supplied on this month's disc for your perusal. **RU**

REAL TIME SOLIDS MODELLER

Silicon Vision has now released its rewritten Real Time Solids Modeller for the Archimedes. Mike Williams investigates.

Silicon Vision is a company which has already established itself in the CAD field with its software for the BBC micro. Now the company has released a completely new version of its Real Time Solids Modeller for the Archimedes.

The large and impressive packaging contains just a single disc and a 45 page manual. In fact, the software consists of two separate components, the SolidCAD Design System for creating and editing 3D objects, and the Real-Time Graphics Language for animating 3D objects. These can be purchased separately if required. We will deal first with the design system.
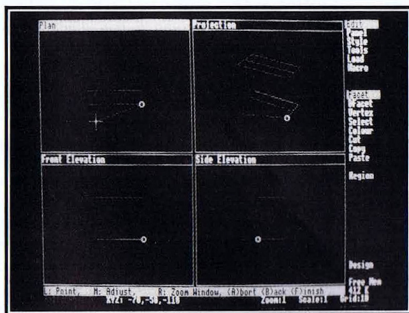
Constructing a simple 3D object

As with other 3D CAD systems (Euclid from Ace Computing for example - see review in RISC User Volume 1 Issue 8), the SolidCAD Design System shows four separate views on the default screen: plan, front view, side view and 3D projection. There is also a menu to the right of the graphics area, and a two line display of information at the foot of the screen.

The main menu list consists of Edit, Panel, Style, Tools, Load and Macro. When any main menu option is selected a subsidiary menu also appears at the right of the screen, often including relevant parameters which can be changed with the mouse.

The software is very largely mouse controlled, but SolidCAD also responds to a number of single key commands, and so-called command line options for controlling the drawing environment. Although it might be felt that a wholly mouse controlled environment would be better, it works well in practice, and has the advantage that the commands can be used to create descriptions of drawings using a word processor, and these descriptions may then be EXECed into the package.

As with any 3D system it takes quite some time to become familiar with the four screen views, and their relationships to each other. The X-Y plane is used for the front elevation, with the Z axis providing depth (and both 1st and 3rd angle projections are supported). All drawing takes place in one of the three 2D views, and is immediately reflected in the display of the other views. All four views can be zoomed in and out, and the image translated up, down, left or right. The 3D projection can also be rotated about any of the three axes. The current state of all these parameters can be displayed by selecting the Panel option from the main menu.

CREATING 3D OBJECTS

Objects are built up from facets (faces) that can be drawn in any plane view using a rubber-banding technique. Closed objects can be automatically completed by pressing 'F'. In addition, any face can be drawn to form the basis of a solid by *extrusion* or by *rotation*. Extrusion forms a solid using the face as a cross-section. The user determines the direction and extent of the extrusion. Perhaps the simplest example is the extrusion of a square to form a cube.

Solid objects can also be created by rotating a plane face about an axis. The user can control both the step size to be used and whether a full or partial rotation is required. With a suitable cross-section, a torus (or doughnut shape) can be readily formed for example.

COLOURING AND SHADING

The Style menu controls the way in which objects are displayed on the screen. The

default is for a complete wire frame object, which has the merit of being fast. The Surface option shows a similar wire frame view, but with hidden line removal . A number of additional options (catering for both convex and concave shapes) allow all the faces of 3D objects to be coloured, giving a really solid appearance.

SolidCAD can be used in mode 12 (16 colours) or mode 15 (256 colours), and of course with multi-sync modes 20 and 21, and the current drawing colour selected from a colour menu. There is also a *smooth* shading option, which introduces shades of the selected colour, but I found considerable trial and error seemed to be necessary, adjusting the various lighting parameters, to obtain a pleasing result. Smooth shading transforms a facetted object into one with much smoother contours. The process only works sensibly in mode 12 if a grey tone set is selected (:GREYMAP), but similar monochrome shadings can be produced in other colours (:BLUEMAP, :REDMAP etc.). Mode 15 (and modes 20 and 21) give true colour shading.

MACROS
A macro allows an object defined on the screen to be saved for future use. The object can then be rescaled and rotated, and positioned as required. Thus a complete display can be created from a variety of separate *building blocks,* each created and saved separately.

REAL-TIME ANIMATION
The Real-Time Graphics Language is supplied as a module which you must load before use. This provides over 40 SYS calls (SWI calls for machine code programmers) which can be used in your own Basic programs. Silicon Vision also provide four demo programs on the disc. The SYS call format may be less familiar to many Basic programmers, but in practice provides simpler and more flexible parameter passing than, say, star commands.

The SYS calls in effect allow the selection of projection (viewpoint) and style of presentation to be changed continuously from within a program to produce an animated

display. This has a lot of potential, and is the subject of further development by Silicon Vision in the form of a Film Maker package.
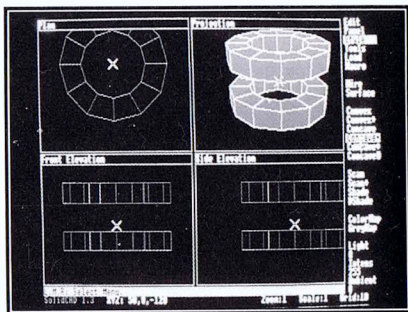
Using the sweep option

DOCUMENTATION
I am sure that any software package that is concerned with creating, editing and manipulating 3D objects will take much practice to use with ease. Thus the documentation has a vital part to play with any such product, particularly for those less used to 3D design.

The original manual is quite brief in its coverage of the software, but is now, fortunately, to be supplemented by a much more extensive tutorial manual with extra examples, as a result of feedback from existing users. This will be supplied as part of the package for all future purchasers, and existing purchasers who have registered with Silicon Vision will receive a copy free of charge. No copy was available at the time of wriring this review.

Silicon Vision also relies on a number of 'README' files on disc to document the latest features of the package. While this is an excellent way of ensuring that a purchaser has the latest software features (and information on their use), I feel that too much reliance has been put on this form of documentation. Even the inclusion of photo-copies of these files would make the information they contain more readily accessible.

The documentation of the Real-Time Graphics language (part of the original manual) gives no simple example programs to show how different SYS calls could be combined to produce worthwhile animation effects. Finally, many of the available keyboard commands appear only in a two page summary at the end of the manual (and many others don't appear at all). There is also, regretably, no index to the existing manual. The good news is that Silicon Vision says that a new version of this manual will be produced when current stocks are exhausted.

CONCLUSIONS

This software provides a powerful 3D modelling system, following as it does several year's of similar software development on the BBC micro. And I have certainly not mentioned all the facilities available. I do not feel however, admittedly on a relatively short trial, that the software is not quite as easy to use as it might be, and the lack of better examples on disc inhibits rather than encourages progress in the early stages of learning.

The documentation, which until now has been less than adequate, looks certain to be improved substantially in the very near future. This is good news for all users, as without better documentation, those new to 3D design in particular would find learning the use of the Real Time Solids Modeller harder than it need be.

The Real-Time Solids Modeller is certainly a powerful piece of software, which is set to reach a much wider audience with the advent of much improved documentation.

| Product	Real Time Solids Modeller
Supplier	Silicon Vision
	Signal House, Lyon Road,
	Harrow, Middx HA1 2AG.
	Tel. 01-422 2274, 01-861-2173
Price	£89.95 Inc. VAT
	(SolidCAD only £49.95, upgrade to
	Real Time Graphics Language £40.00)

RU

Archimedes Visuals

A Sixteen-Colour Fader and a Textured Spirograph

SIXTEEN-COLOUR FADER
by Ledger White

This program contains procedures which allow you to fade any image up and down in the 16-colour modes.

If you run the program, a sequence of randomly coloured ellipse-based patterns will be displayed. Each will fade up from black, and then fade back down before the next appears. Although the patterns are pleasant to look at, they are only incidental, and serve simply to illustrate the fades.
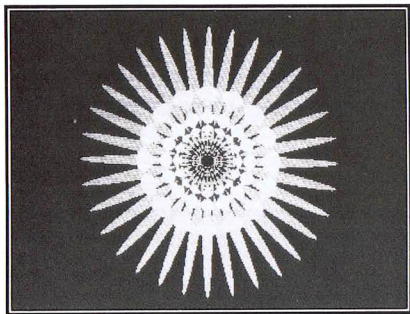
Pattern used by the fader program

To use the fade procedures in your own program, you will need to replace the two procedures PROCscreenPalette and PROCgeneratePattern. The first establishes the palette which will be used with the display. In the present example, a random palette is used. Your version of PROCscreenPalette should set up your own palette. The second procedure places the image on the screen. In the current program, a random pattern is drawn, but you might replace this with a *ScreenLoad to load a pixel-based image, or whatever.

The program's three procedures PROCstorePalette, PROCfadeIn and PROCfadeOut do all the work. The first stores the current palette in the array col%(i,j), and then sets all physical colours to black. PROCfadeIn brings the palette up from black to the stored values, while PROCfadeOut takes it back to black.

```
   10 REM            >Fader
   20 REM Program    Smooth Screen Fade
   30 REM Version    A 0.03
   40 REM Author     Ledger White
   50 REM RISC User March 1989
   60 REM Program    Subject to Copyright
   70 :
   80 ON ERROR MODE 9:PRINT REPORT$;" @
line ";ERL:END
   90 DIM col%(15,2)
  100 MODE 9:OFF
  110 X=RND(-TIME)
  120 REPEAT
  130   PROCscreenPalette
  140   PROCstorePalette
  150   PROCgeneratePattern
  160   PROCfadeIn
  170   PROCwait(200)
  180   PROCfadeOut
  190 UNTIL FALSE
  200 :
  210 DEFPROCscreenPalette
  220 FOR X%=1 TO 15
  230 COLOUR X%,16*RND(15),16*RND(15),16
*RND(15)
  240 NEXT
  250 ENDPROC
  260 :
  270 DEFPROCgeneratePattern
  280 GCOL 3,RND(15)
  290 Z%=RND(30):M%=450
  300 FOR E=11.25 TO 180 STEP 11.25
  310 ELLIPSE FILL 640,512,Z%,M%,RAD(E)
  320 NEXT
  330 GCOL 3,RND(15)
  340 Z%=RND(100):M%=250
  350 FOR E=22.5 TO 180 STEP 22.5
  360 ELLIPSE FILL 640,512,Z%,M%,RAD(E)
  370 NEXT
  380 ENDPROC
  390 :
  400 DEFPROCstorePalette
```

```
 410 FOR logcol%=0 TO 15
 420 SYS "OS_ReadPalette",logcol%,16 TO
,,palette
 430 col%(logcol%,2)=palette>>>24
 440 col%(logcol%,1)=(palette>>>16) AND
&FF
 450 col%(logcol%,0)=(palette>>>8) AND
&FF
 460 COLOUR logcol%,0,0,0
 470 NEXT
 480 ENDPROC
 490 :
 500 DEFPROCfadeIn
 510 FOR x=0.0625 TO 1.01 STEP 0.0625
 520 WAIT
 530 FOR Y%=0 TO 15
 540 COLOUR Y%,col%(Y%,0)*x,col%(Y%,1)*
x,col%(Y%,2)*x
 550 NEXT
 560 NEXT
 570 ENDPROC
 580 :
 590 DEFPROCwait(delay%)
 600 T%=TIME
 610 REPEAT
 620 UNTIL TIME>T%+delay%
 630 ENDPROC
 640 :
 650 DEFPROCfadeOut
 660 FOR x=0.9375 TO -0.01 STEP -0.0625
 670 WAIT
 680 FOR Y%=0 TO 15
 690 COLOUR Y%,col%(Y%,0)*x,col%(Y%,1)*
x,col%(Y%,2)*x
 700 NEXT
 710 NEXT
 720 CLS
 730 ENDPROC
```

SPIROGRAPH
by Mike Ironmonger

This very short program produces some quite stunning effects. It first creates a small sprite containing the 256 colours of a mode 13 screen. This is then plotted on the screen on the locus of a circle, which is itself turning around the inside of a larger ring. Parameters are chosen randomly, and each press of the space bar creates a new pattern. A very wide variety of effects is generated, though no screenshots will do them justice, because even

if they convey the texture of a particular pattern, they can give no idea of the rippling movements created as the program runs.

Example spirograph

```
  10 REM          >SpiroGrph
  20 REM Program   Spirograph Visuals
  30 REM Version   A 1.00
  40 REM Author    Mike Ironmonger
  50 REM RISC User March 1989
  60 REM Program   Subject to Copyright
  70 :
  80 MODE 13:OFF
  90 X%=0:Y%=0
 100 FOR C%=0 TO 63
 110 FOR T%=0 TO 3
 120 GCOL C% TINT T%*64
 130 POINT X%,Y%
 140 Y%+=4:IF Y%=64 Y%=0:X%+=4
 150 NEXT:NEXT
 160 MOVE 0,0:MOVE 60,60
 170 *SGET block
 180 *SCHOOSE block
 190 :
 200 ORIGIN 610,482
 210 REPEAT
 220 CLS:S%=RND(200)
 230 A=0:B=PI*RND(1):C=0:D=B*RND(1)
 240 REPEAT
 250 X%=250*SIN(A):Y%=250*COS(A)
 260 PLOT &ED,X%+S%*SIN(C),Y%+S%*COS(C)
 270 A+=B:C+=D
 280 UNTIL INKEY$(0)<>""
 290 UNTIL FALSE
```

David Spencer completes the current Toolbox series with four final commands.

This month we add commands to search disc files for a given byte, 32-bit word, or character sequence; and also a useful help command.

ENTERING THE PROGRAM

This month's listing should be entered in exactly the same way as for previous ones, ensuring the program has not been renumbered. Save the source code program and run it. This assembles and saves the Toolbox module which can then be loaded by typing QUIT followed by TOOLBOX.

HELP

The first new command is invoked using *TOOLS, or *HELP TOOLS. This lists out all the commands that the *RISC User Toolbox* accepts. More details of each command can be found using *HELP <command name>.

DISC SEARCHING

There are three commands for searching disc files. These are:

```
*DSEARCHB <byte>... <filename>
          [<pathname>] [V][C]
*DSEARCHW <word>... <filename>
          [<pathname>] [V][C]
*DSEARCHS <string> <filename>
          [<pathname>] [V][C]
```

They are used to search for byte sequences, 32-bit word sequences and character strings respectively.

All these commands have a similar syntax. Each starts with the search data, which is either a series of bytes or 32-bit words (both in hex) separated by spaces, or a character string enclosed in quotes for *DSEARCHS. Next is a wildcarded filename that specifies which files to search, and an optional pathname to show where the search is to start. Finally, there can be up to two qualifiers:

V   Display name of each file as it is searched.
C   Ask for confirmation before searching a file.

The commands will search the directory tree from the starting point specified, examining files that match the wildcarded filename given. If the

C option is used, you are asked whether each match should be searched or not. You can also at this point choose to stop confirming for each file by pressing Q (for Quiet), or to abort the search by pressing A. As each file is searched, it is loaded into application memory. Therefore, any program in memory is lost, and should be saved first.

Here are some examples of the use of these commands:

```
*DSEARCHB 9F 5 1A "EDIT" $.UTILS
```
to search for the byte sequence 9F,5,1A in the file $.UTILS.EDIT. Note the use of quotes to stop the 'ED' of 'EDIT' being interpreted as a fourth byte to search for.

```
*DSEARCHW FFFF 0000 PROG* ADFS:$ VC
```
to search for the word sequence FFFF, 0000 in all files with names beginning 'PROG' on the current disc. Additionally, confirmation is asked before searching each file, and the name of each file is printed as it is searched.

```
*DSEARCHS "Hello" * "" C
```
to search for the word 'Hello' in all files starting from the current directory. Again, confirmation is sought before searching each file.

Note that if any qualifiers are given without a pathname, then "" must be used to show that it has been omitted, as in the last example.

For each match found, the name of the file is printed along with the position of the match in hex. Escape can be used to stop the search at any time, although it might take a few seconds before the program exits.

RISC OS

A number of minor changes are necessary in order to make the *RISC User Toolbox* function correctly with RISC OS. These will be covered in a future update which we expect to include in the next issue.

LONG LIVE TOOLBOX

We have come to the end of the *RISC User Toolbox* for the time being. However, its flexible design means that further updates can be added at a later date, and we would still like to hear from members with any ideas which they would like to see included.

```
  251 SWI "OS_WriteS"
  252 EQUS "The RISC User Toolbox has be
en installed.":EQUW &D0A
  253 EQUS "Use *TOOLS or *HELP TOOLS fo
r a list of commands.":EQUW &D0A:EQUB 0
  261 .gettaba
  262 ADR R3,tab:MOV PC,R14
  359 EQUS "Tools":EQUB 0
  360 ALIGN:EQUD toolc:EQUD &20FF0000
  361 EQUD 0:EQUD toolc
  362 EQUS "DSearchB":EQUB 0
  363 ALIGN:EQUD dsb:EQUD &FF0002
  364 EQUD dsbsyn:EQUD dsbhlp
  365 EQUS "DSearchW":EQUB 0
  366 ALIGN:EQUD dsw:EQUD &FF0002
  367 EQUD dswsyn:EQUD dswhlp
  368 EQUS "DSearchS":EQUB 0
  369 ALIGN:EQUD dsc:EQUD &FF0002
  370 EQUD dscsyn:EQUD dschlp
 1448 .dsbhlp EQUS "*DSearchB searches s
elected disc files for the given byte se
quence.":EQUB 13
 1449 .dsbsyn EQUS "Syntax: DSearchB <by
te>... <filename> [<pathname>] [V] [C]":
EQUB 0:ALIGN
 1450 .dswhlp EQUS "*DSearchW searches s
elected disc files for the given 32-bit
word sequence.":EQUB 13
 1451 .dswsyn EQUS "Syntax: DSearchW <wo
rd>... <filename> [<pathname>] [V] [C]":
EQUB 0:ALIGN
 1452 .dschlp EQUS "*DSearchS searches s
elected disc files for the given charact
er string.":EQUB 13
 1453 .dscsyn EQUS "Syntax: DSearchS <st
ring> <filename> [<pathname>] [V] [C]":E
QUB 0:ALIGN
 6870 CMP R3,#4
 7210 .se3 ADD R4,R12,#256:BL swi1:BEQ s
e5
14870 BNE nohash:CMP R3,#0:MOVEQ R3,#1
14871 MOVNES R3,#0:B ma
14872 .nohash EOR R3,R2,R3
14880 AND R3,R3,#&DF:.ma CMP R3,#0
15480 :
15490 .toolc STMFD R13!,{R14}
15500 MOV R0,#&75:SWI "OS_Byte"
15510 STMFD R13!,{R1}:SWI &10E
15520 SWI "OS_NewLine"
15530 SWI "OS_WriteS"
15540 EQUS "The RISC User Toolbox module
 offers the following commands:"
15550 EQUW &D0A:EQUB 0
15560 BL gettaba
15570 .toolc2 MOV R0,R3:SWI "OS_Write0"
15580 ADD R3,R0,#19:BIC R3,R3,#3
15590 LDRB R0,[R3]:CMP R0,#0
15600 BNE toolc25:SWI "OS_NewLine"
15610 LDMFD R13!,{R1}:TST R1,#4
```

```
15620 SWIEQ &10F:SWI "OS_WriteS"
15630 EQUS "Use *HELP <command> for deta
ils of individual commands."
15640 EQUW &D0A:EQUB 0:LDMFD R13!,{PC}
15650 .toolc25 SWI &120
15660 MOV R0,#&86:SWI "OS_Byte"
15670 CMP R1,#101:SWICS "OS_NewLine"
15680 BCS toolc2
15690 .toolc3 SUBS R1,R1,#20
15700 BCS toolc3
15710 .toolc4 SWI &120
15720 ADDS R1,R1,#1:BNE toolc4
15730 B toolc2
15740 :
15750 .dsw MOV R3,#4:B dsb2
15760 .dsb MOV R3,#1
15770 .dsb2 STMFD R13!,{R14}
15780 LDR R12,[R12]:ADD R5,R12,#&400
15790 .getseq MOV R1,R0:MOV R4,#16
15800 .getseq2 MOV R0,#16
15810 CMP R3,#4:ORRNE R0,R0,#1<<30
15820 ORR R0,R0,#1<<31
15830 SWI "OS_ReadUnsigned"
15840 STR R2,[R5,R4]:ADD R4,R4,#4
15850 SUB R1,R1,#1
15860 .getseq3 ADD R1,R1,#1
15870 LDRB R2,[R1]:CMP R2,#32
15880 BEQ getseq3:CMP R2,#&30
15890 BCC getseq4:CMP R2,#ASC"G"
15900 BCS getseq4:CMP R2,#&3A
15910 BCC getseq2:CMP R2,#ASC"A"
15920 BCS getseq2
15930 .getseq4 STR R4,[R5,#4]
15940 CMP R2,#32:ADRCC R0,nofile
15950 LDMCCFD R13!,{R14}
15960 ORRCCS PC,R14,#1<<28
15970 .dscont MOV R0,#0
15980 STRB R0,[R12,#&402]
15990 STMFD R13!,{R1}:SWI "OS_GetEnv"
16000 SUB R1,R1,#&8000
16010 STR R1,[R12,#&408]
16020 LDMFD R13!,{R1}:MOV R0,R1
16030 MOV R2,#1<<29:SWI "OS_GSInit"
16040 ADD R5,R5,#512
16050 .gfnn SWI "OS_GSRead"
16060 MOVCS R1,#0:STRB R1,[R5],#1
16070 BCC gfnn:SUB R0,R0,#2
16080 TST R2,#1<<28:SUBNE R0,R0,#1
16090 .gfnn2 LDRB R1,[R0,#1]!
16100 CMP R1,#32:BEQ gfnn2:MOV R4,R0
16110 MOVCC R2,#0:BCC gopdone
16120 MOV R2,#1<<29:SWI "OS_GSInit"
16130 .skpath SWI "OS_GSRead":BCC skpath
16140 TST R2,#1<<28:SUBNE R0,R0,#1
16150 MOV R2,#0:SUB R0,R0,#1
16160 .gop LDRB R1,[R0],#1:CMP R1,#32
16170 BEQ gop:BCC gopdone:AND R1,R1,#&DF
16180 CMP R1,#ASC"V":ORREQ R2,R2,#1
16190 BEQ gop:CMP R1,#ASC"C"
```
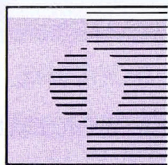
```
16200 ORREQ R2,R2,#2:BEQ gop:B bqual
16210 .gopdone STMFD R2,[R12,#1024]
16220 STRB R3,[R12,#1025]
16230 ADR R0,getout:STMFD R13!,{R0}
16240 STR R13,[R12,#&3FC]
16250 MOV R0,R4:ADR R1,serrou
16260 BL swi11:ADD R13,R13,#4
16270 .getout LDRB R0,[R12,#&402]
16280 CMP R0,#0:LDMEQFD R13!,{PC}^
16290 SWI "OS_Exit"
16300 .dsc STMFD R13!,{R14}
16310 LDR R12,[R12]
16320 MOV R2,#1<<29:SWI "OS_GSInit"
16330 ADD R3,R12,#256
16340 .dsc2 SWI "OS_GSRead"
16350 MOVCS R1,#0:STRB R1,[R3],#1
16360 BCC dsc2:TST R1,#&28
16370 SUBNE R0,R0,#1:SUB R0,R0,#2
16380 STMFD R13!,{R0}
16390 ADD R0,R12,#256
16400 ADD R1,R12,#&410:BL swi2
16410 LDMFD R13!,{R1}:MOV R3,#0
16420 .ssss LDRB R0,[R1,#1]!:CMP R0,#32
16430 BEQ ssss:B dscont
16440 :
16450 .serrou STMFD R13!,{R0-R1}
16460 ADD R0,R0,#20:ADD R1,R12,#&600
16470 BL swi13:CMP R3,#0
16480 LDMNEFD R13!,{R0-R1,PC}
16490 ADD R2,R12,#255
16500 .concat LDRB R3,[R2,#1]!
16510 CMP R3,#0:BNE concat
16520 STMFD R13!,{R2}:ADD R3,R12,#256
16530 CMP R2,R3:BEQ nodot
16540 LDRB R3,[R2,#-1]:CMP R3,#ASC":"
16550 MOVNE R3,#&2E:STRNEB R3,[R2],#1
16560 .nodot
16570 LDR R0,[R13,#4]:ADD R0,R0,#20
16580 .concat2 LDRB R3,[R0],#1
16590 CMP R3,#0:STRB R3,[R2],#1
16600 BNE concat2:LDR R0,[R13,#4]
16610 LDR R1,[R0,#16]:CMP R1,#2
16620 BEQ nosthis:LDR R1,[R0,#8]
16630 LDR R0,[R12,#&408]:CMP R1,R0
16640 BCC lenok:SWI "OS_WriteS"
16650 EQUS "File ":EQUB 0
16660 ADD R0,R12,#256
16670 SWI "OS_Write0":SWI "OS_WriteS"
16680 EQUS " too long - skipping."
16690 EQUW &D0A:EQUB 0:B nosthis
16700 .lenok STR R1,[R12,#&40C]
16710 LDRB R0,[R12,#&400]:TST R0,#2
16720 BEQ noconf:SWI "OS_WriteS"
16730 EQUS "Search file ":EQUB 0
16740 ADD R0,R12,#256:SWI "OS_Write0"
16750 SWI "OS_WriteS"
16760 EQUS " (Y/N/Q/A) ? ":EQUB 0
16770 .kkget SWI "OS_ReadC"
16780 BCS abort:AND R0,R0,#&DF
```

```
16790 CMP R0,#ASC"N":BLEQ pnl
16800 BEQ nosthis:CMP R0,#ASC"Y"
16810 BLEQ pnl:BEQ noconf
16820 CMP R0,#ASC"A":BLEQ pnl
16830 BEQ abort:CMP R0,#ASC"Q"
16840 BNE kkget:BL pnl
16850 LDRB R0,[R12,#&400]
16860 BIC R0,R0,#2:STRB R0,[R12,#&400]
16870 .noconf LDRB R0,[R12,#&400]
16880 TST R0,#1:BEQ notell
16890 SWI "OS_WriteS"
16900 EQUS "Searching file ":EQUB 0
16910 ADD R0,R12,#256:SWI "OS_Write0"
16920 SWI "OS_NewLine"
16930 .notell ADD R1,R12,#256
16940 MOV R2,#&8000:MOV R3,#0
16950 MOV R0,#&FF:SWI "OS_File"
16960 MOV R0,#1:STRB R0,[R12,#&402]
16970 LDR R2,[R12,#&40C]
16980 MOV R0,#&8000:ADD R1,R2,R0
16990 LDR R2,[R12,#&404]
17000 SUB R2,R2,#16:MOV R2,R2,LSR #2
17010 LDRB R3,[R12,#&401]
17020 CMP R3,#4:BICEQ R1,R1,#3
17030 CMP R3,#0:BEQ smsl
17040 ADD R4,R12,#&410
17050 .msl BL swi1:BNE nosthis
17060 BL fndprt
17070 ADD R0,R0,R3:B msl
17080 .smsl LDRB R0,[R12,#&410]
17090 LDR R2,[R12,#&40C]:SUB R2,R2,R0
17100 ADD R2,R2,#&8000:MOV R0,#&8000
17110 ADD R1,R12,#&410
17120 .smsl2 BL swi3:BEQ smsl4
17130 .smsl3 ADD R0,R0,#1:CMP R0,R2
17140 BNE smsl2:B nosthis
17150 .smsl4 BL fndprt:B smsl3
17160 .fndprt STMFD R13!,{R0-R3,R14}
17170 SWI "OS_WriteS":EQUS "Match found
in file ":EQUB 0
17180 ADD R0,R12,#256:SWI "OS_Write0"
17190 SWI "OS_WriteS":EQUS " at offset &
":EQUB 0
17200 LDR R1,[R13]:SUB R1,R1,#&8000
17210 BL adrprt:SWI "OS_NewLine"
17220 LDMFD R13!,{R0-R3,PC}
17230 .abort LDR R13,[R12,#&3FC]
17240 LDMFD R13!,{PC}
17250 .nosthis LDMFD R13!,{R0}
17260 MOV R1,#0:STRB R1,[R0]
17270 LDMFD R13!,{R0-R1,PC}
17280 .pnl STMFD R13!,{R14}
17290 SWI "OS_WriteC":SWI "OS_NewLine"
17300 LDMFD R13!,{PC}^
17310 .nofile EQUD 0
17320 EQUS "Missing Filename"
17330 EQUB 0
```

ARCade

A survey of the games software scene for the Archimedes compiled by Peter Rochford.

When the Archimedes was first launched some eighteen months ago, there was little software around to go with it, least of all games. If you wanted to thrash a few aliens with your 32-bit wonder machine, you had no choice other than *Lander* on the Welcome disc.

The first games release came from Superior Software in the shape of *Zarch* (£19.95), an enhanced version of Lander featuring extra graphics and sound. Following Zarch, Superior then released *Conqueror* (£24.95), a game that owes much to the graphics in Zarch. It features a not dissimilar 3D scrolling landscape, but used now in a tank battle simulation. It was OK if you had a taste for that kind of thing, but the Zarch-like graphics gave that feeling of *deja vu*.

Since those two games were released nothing much has appeared until the last few months, when several new games have surfaced.

Minerva has done much to support the Archimedes with serious software, and they have also released several games too. Their first was *Minotaur* (£14.95), an unremarkable 3D maze-type game that involves plodding round seemingly endless corridors trying to find a way out. Graphically abysmal, and about as exciting as a wet Sunday afternoon. Sorry Minerva!

Their follow-up release was *Missile Control*, a game that appeared in the arcades years back, with versions for many machines including the Beeb. The game involves destroying missiles that travel down the screen, aiming to destroy the cities below. The graphics in this are crude when you consider what can be done on the Arc, and playing the game turns out to be just as much a mega-yawn as Minotaur.

Redemption for Minerva comes in the shape of one of their latest releases, *Orion*. This is a much enhanced version of the classic arcade

game Defender that appeared on the Beeb as Planetoid. It is a horizontally scrolling, shoot-em-up game, where you have to save the little men on the planet's surface as they get abducted by the aliens dropping down from above, at the same time avoiding or destroying a whole host of other nasties. Orion is a brilliant version of the game with a classy presentation, and excellent graphics and sound. Highly recommended!
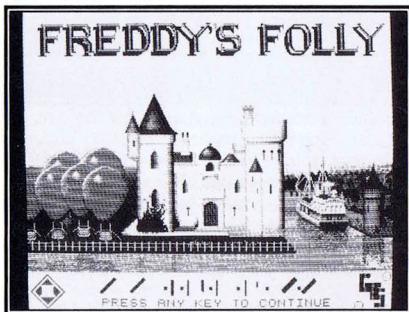
Freddy's Folly

Freddy's Folly is another offering from Minerva, but aimed at the younger Arc user. The game scene is a harbour where your castle is built overlooking the sea. The idea is to prevent your castle from being bombed by the balloons that fly overhead by shooting them down with a cannon. You must also destroy the ships which are shooting at you as they sail into the harbour. The game is preceded by a little story that has some rather amusing animated sequences as illustration. The whole thing is well thought out and nicely implemented, although I thought the text in the story should be left on the screen longer for youngsters to read. Otherwise, a good game with lots of appeal.

The latest Minerva game is *Jet Fighter*, a version of yet another computer favourite that was most popular on the Spectrum under the

name Jet Pac. You have to fly your spaceman around the screen collecting parts to construct a space rocket that will transport you to the next level for more of the same. It's pretty mundane stuff, with graphics comparable to versions of this game on lesser machines.

Impact Games' first and as yet only release for the Arc is *Quazer,* a frantically fast, vertically scrolling, death and destruction shoot-em-up. The game makes good use of graphics with hordes of colourful sprites hurtling down the screen accompanied by a host of noisy sound-effects. A thoroughly enjoyable and frenetic alien bash, with lasting appeal and definitely recommended.

Dabs Press has now entered the games software market with *Alerion* for the Arc. Like Quazer, this is a vertically scrolling shoot-em-up that is frantic and noisy. The graphics are good, and it features digitised sound effects too. My own personal preference between the two similar games would be Quazer. But Alerion is challenging and enjoyable. Worth considering if you like this type of thing.

A totally different kind of offering from Dabs is *Arcendium,* a collection of computer versions of four popular board games, Backgammon, Reversi, Draughts and Quadline (Connect-4). These are nicely written, featuring some neat graphics, and are played under mouse control. If you like board games then Arcendium is worth buying, but be warned, the computer plays a mean game!

Gem Electronics, a new name in the games software world, has released two Arc games to date. The first is *Star Trader,* a strategic space adventure that can be likened in some ways to Elite, and shares many of its features. However, Elite it is not, and the combat elements in particular do not in any way compare. It is a brave attempt though, and is an involved game that can be entertaining and absorbing to play.

The other release from Gem is *Word-Up-Word-Down,* a 3D scrabble game that really is

excellent. It has delightful graphics and it all works beautifully, mostly under mouse control. The program features a 30,000 word dictionary and digitised speech. Up to six people can play, or you can play against the computer: highly recommended.

Just when you thought that Repton had at long last been laid to rest with the final(?) *ultimate* Repton Infinity on the Beeb, up it pops again for the Arc as *Repton 3.* If you like Repton, all well and good, but this is just identical to the Beeb versions, and to my mind an unwarranted release from Superior.
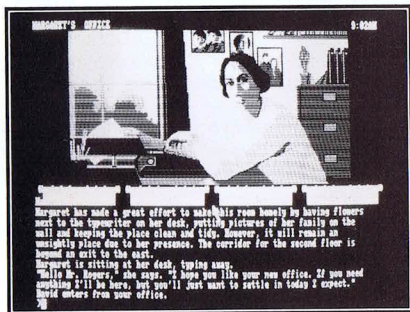
Corruption

Rainbird is a software division of the great British Telecom. They have now entered the Arc games market, and it is encouraging that one of the 'biggies' has sat up and taken notice of the machine. They have released two adventure games. The first one is *Corruption* (£24.99), set in yuppie-land where you take the role of a partner in a broking firm. The plot is in the name of the game - corruption.

The second game is called *Fish!* (£24.99), with an unbelievably zany plot. You are a goldfish and your first task is to escape from your bowl. Having done so, you are transformed into a fish scientist, Mr A. Roach, and from then on you are the leader of The Project, a secret undertaking to conserve

water. You must prevent The Project from being sabotaged. Crazy!

Space does not permit a detailed review of these games but take it from me, they are both brilliant. From the amazing opening music, to the superb graphics and the game-play itself, it is all first class stuff. Buy them both!
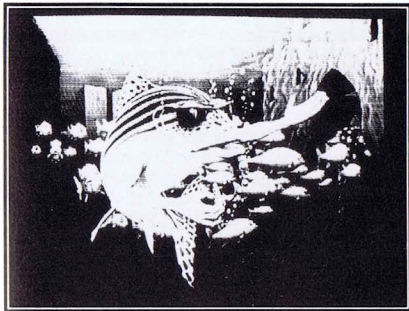
Fish!

Finally, two games from a company called Grand Slam. *Pacmania* is a 3D version of the old favourite Pacman. You may stop yawning at the thought of more Pacman! Pacmania is a whole new slant on the original, and features probably the best graphics and music yet on the Arc. There are four different 'worlds' to negotiate, at ever higher levels of difficulty. The animation is truly brilliant from the superb 4-way screen scrolling to the amusing cartoon-like features that appear between each level.

Terramex is the other release from Grand Slam. This is an animated graphic adventure where you have to travel across deserts and through caves, collecting items to help you in your search for the mad scientist, who is the only person who can save the world from being destroyed by a meteorite. The plot may be a bit corny, but the game is delightful and amusing. It is also very challenging, and this reviewer is still struggling! The graphics are truly excellent and the game is incredibly addictive. You must forgive my gushing enthusiasm for both Pacmania and Terramex, but they are of the

quality Arc owners have been waiting for, and do some justice to the machine itself at long last. Need I say more!

Terramex

COMMENT

In this survey, I have only looked briefly at most of the games yet released for the Arc. In the future I hope to provide f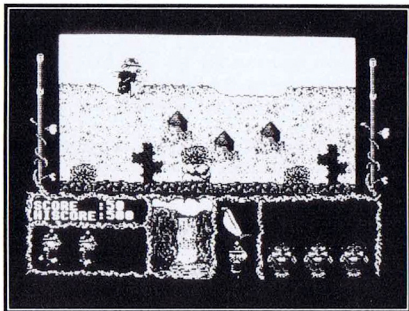uller reviews of most major games as soon as they are released. There are now good signs that the games scene for the Arc is improving at long last. Games like the Firebird and Grand Slam releases are converted versions of Amiga and ST software which is good. If they sell well enough I am sure that many more will follow. Anything you can do on the ST and Amiga you can do just as well or, better on the Archimedes.

Which brings me to my final point. As good as Pacmania or Terramex are, and they are the best around for the Arc, they still do not exploit the full potential of the machine's speed, graphics and sound capabilities. The true fact is, that until the Arc has sold in even greater numbers, sadly no one is going to invest the time and money in a mega-game that does. But the Arc is selling better than ever now, and the future is looking brighter all the time. We live in hope. See you soon!

NOTE: All games referred to in this survey are available from BEEBUG, and at 5% discount to RISC User members - see retail catalogue. RU

**David Spencer concludes our introduction to the RISC OS multi-tasking system
by looking at it from a programmer's point of view.**

Before explaining how the multi-tasking system works, we need to understand the interaction between the Window Manager and a program running in a WIMP environment. The key to this is a technique called *polling*.

The program must regularly call a routine in the Window Manager (via a SWI call) known as the polling routine. The Window Manager uses this opportunity to perform any of its own processing, such as redrawing a drag box, and then returns a *reason code* to the program. This reason code informs the program of any actions that the Window Manger requires it to perform. For example, if a reason code of 3 was returned, then this would mean that the user had clicked on the 'Quit' box of an open window. Before performing any other actions, the program must find out which window is to be closed, and ask the Window Manager to close it.

Similarly, a reason code of 1 means that a window needs redrawing (perhaps because a window that was on top has been closed), while a reason code of 8 signifies that the user has pressed a key. A reason code of 0 means that the Window Manager doesn't want any actions performing at that point, and the program is free to do any processing of its own, for example updating the display. Whatever the reason code returned from the polling routine, as soon as the program has performed the necessary action, it must return to the loop that performs the polling.

The net outcome of the polling system is that control is bounced backwards and forwards between the application program and the Window Manager. To achieve multi-tasking all that is necessary is for the Window Manager to pass control to each application in turn. This is shown graphically in the diagram, which illustrates the multi-tasking of six applications. We will take as a starting point the situation where *App.1* is about to call the polling routine. *App.1* calls the Window Manager, much in the same way as would happen in a single task

situation. However, when the Window Manager has finished its processing, rather than returning to *App.1*, it instead passes control to *App.2*, which performs any necessary actions and duly calls the polling routine again. Subsequently, control is passed to *App.3*, and then *App.4* and so on. Finally, when *App.6* calls the polling routine, the Window Manager will pass control back to *App.1*, with whatever reason code applies to *App.1* at that time.

As far as *App.1* is concerned, it has called the Window Manager's polling routine, and the Window Manager has eventually returned a reason code back to it. *App.1* is totally unaware of the fact that six other applications have been activated one after another, while it was waiting for the routine to return. Had we considered how any of the other five applications viewed what was happening, the situation would be unchanged. As far as each application is concerned, it is the only one running.

Therefore, we have achieved our aim of a multi-tasking system with very little effort. It should also be clear now why the Window Manager plays such an important role in multi-tasking, and why all multi-tasked applications run in a WIMP environment.

MEMORY ALLOCATION

In a multi-tasking system it is important to control the allocation and use of memory. All applications running under RISC OS expect to have a contiguous area of RAM starting at address &8000 to use as workspace. In the case of applications loaded from a file, they actually load at this address and then use the memory above them as workspace. Module based applications, such as Basic, are free to use all the application memory as workspace.

The amount of application workspace available overall is not set directly. Instead, all the RAM left over after the areas have been allocated for the screen, modules, sprites, etc. is used as application workspace. In a single-tasking environment, only one application is

active at a time, and consequently, all the application workspace is allocated to this application.
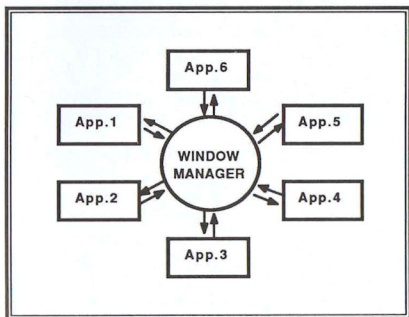
The WIMP polling system

In a multi-tasking environment, the situation is more complicated. The *Task Manager* maintains a pool of free memory, out of which, memory can be allocated to tasks. A certain quantity of this memory is made available to the next application that is started. The user can change this allocation via the task display window. When a new application is loaded it uses a star command to tell the task manager the minimum amount of memory it requires, and the maximum that it could use if more was available. Provided that the memory set aside for the next task is enough, the application will be started and allocated the appropriate amount of workspace. Some applications can change the amount of memory they have while they are running. They do this by claiming more memory from the free pool when needed, and returning it when it is no longer required. An example of this type of application is ArcEdit on the Welcome disc. When an application is removed, its memory is returned to the free pool.

It is all very well to allocate self contained blocks of memory to each application, but there still remains the problem of ensuring that whenever an application is actually executing, its workspace lies in a block from location &8000 upwards. One obvious way would be to copy the workspace of the current application to somewhere else, copying the workspace for the next application down to &8000, and then finally executing the next application. However, even with the speed of the ARM, copying a large amount of memory wastes a lot of time.

Luckily, MEMC, the ARM memory controller chip, provides a far more elegant solution. Instead of mapping addresses directly to locations within the physical RAM, MEMC supports a concept known as *logical memory*. The actual physical memory is split up into 128 *pages*. Therefore, on a 310, each page is 8K long, while on a 440 it is 32K. A 305 keeps a page size of 8K and only uses 64 pages, rather than having a 4K page size. For each page of physical memory, it is possible to define where it appears in the memory map of the ARM processor. The mapping between the logical processor addresses and the actual physical addresses is controlled by an area of very fast memory within MEMC called a *Content Addressable Memory* (CAM). It is by using this technique that RISC OS arranges for the areas of RAM allocated to the screen, sprites, etc. to start at well-defined addresses.

However, the ability to change the address at which a page of memory appears is perfectly suited to the task of moving application memory. When switching from one application to another, RISC OS changes the contents of the CAM so that the current application's workspace is remapped to an address that is out of the way of anything else. The RAM pages that belong to the next application are then located using a table kept by the Window Manager, and the CAM is further changed so that these pages are mapped from address &8000 upwards. This achieves the effect of moving around large amounts of memory by only altering a few locations within the CAM. Furthermore, MEMC also allows pages of memory to be protected, and RISC OS ensures that all application memory not in use at that time is protected so that it cannot be written to. This means there is less likelihood of one application accidentally corrupting the workspace belonging to another.

This ends our introductory look into the world of multi-tasking, but we will return to the subject in future issues of RISC User. **RU**

Clares' Toolkit Plus

Reviewed by David Spencer

The original *Toolkit* from Clares was one of the first pieces of Archimedes software to be released, and has therefore been around for nearly two years. *Toolkit Plus* is the long promised, enhanced version of *Toolkit*, and it is this which I will look at here.

EDITORS

In common with most Toolbox-type packages, *Toolkit Plus* is based around a full screen editor that can edit either memory or disc contents. The editor can be invoked directly using the *MEDIT and *AEDIT commands, or indirectly via the search commands described below. The editor display can be switched between binary, hex, word, text or mnemonic format, and the display freely scrolled. One very nice feature is the single-line assembler that allows ARM mnemonics to be assembled directly into memory from within the editor.

In addition to the interactive editor display, blocks of memory can also be dumped in any of the available display formats. This is mainly used when dumping memory to a printer.

FIND COMMANDS

Toolkit Plus includes a large number of search and find commands that can be used to search either memory or disc for a given sequence of bytes, words, characters or mnemonics. When a match is found, the editor is entered, enabling changes to be made before the search continues.

MISCELLANEOUS

There are a number of commands which don't fit into any obvious category. These include commands to move and swap blocks of memory, and to list out various important system addresses, such as that of the current error handler.

Also in this group are commands to list all the SWI calls available in the machine, and to convert a SWI name to its corresponding number, and vice-versa. These can all be very useful, but in some cases only reproduce what the operating system already offers.

GENERAL FILING SYSTEM COMMANDS

As well as the disc editor and disc searches, *Toolkit Plus* also provides some general file-specific commands.

*CATALL, *EXALL and *DIRALL all perform a cataloguing type function. *CATALL lists all the files and sub-directories starting from a given place, while *EXALL also lists full information on those files. *DIRALL on the other hand only lists directory names.

*APUT and *AGET are used to read and write disc sectors directory. They can be useful when recovering lost files, or writing a disc protection system, though these calls really only provide a star command equivalent to an SWI call provided by the ADFS.

There is a command to compact an ADFS disc repeatedly until only one chunk of free space remains, and another command to verify the contents of a file against a block of memory. Finally in this category, is a very useful command to dump to screen (or printer) the contents of a file in any of binary, hex, word, text or mnemonic format.

SCREEN HANDLING COMMANDS

The last set of commands is connected with screen handling. The first two of these are *FASTSAVE and *FASTLOAD for saving and loading screens. These commands are much faster than *SCREENSAVE and *SCREENLOAD, but only work for whole screens. This seems more limiting than necessary, because the method used by these commands will in fact work with any window that is the full screen width, rather than just full screens. The commands *LOADPALETTE and *SAVEPALETTE save and load the current palette setting.

HARD DISC ARCHIVER

This is a utility for backing up hard discs, which is effectively separate from the main commands, although it is invoked using *Toolkit Plus* commands. The archiver is in fact a Basic program which copies the contents of a hard disc onto a set of floppies (as many as 27 of

them for a full hard disc). The copy is done as a series of disc sectors, rather than on a file by file basis, and this removes the restriction of not being able to back up files longer than 800K. The resultant backup consists of two parts: the actual data that makes up the files, and a backup log showing where the data belongs on the hard disc. This log is written twice to separate floppies for extra protection.

Various configuration options can be used to determine which files will be archived. It is possible to archive only files before a certain date/time, or after a certain date/time, or inside or outside a particular period of time. You can also choose to archive files of just one particular filetype.

The opposite of *Archive is *Restore, which restores all the archived files back to the hard disc after any disaster. All files must be restored at once, it is not possible to restore just selected files.

CONFIGURATION

Toolkit Plus makes extensive use of the *CONFIGURE command to customise its operation. For example, default pathnames and starting addresses can be set up. Unlike the normal configuration options, *Toolkit Plus* doesn't save them in CMOS RAM. Instead, the options modify the program directly, with an option to save them to disc as permanent changes.

DOCUMENTATION

The *Toolkit Plus* manual is very similar to that of *Toolkit*, and indeed any other of Clares' products. It is A5 in size, 64 pages long, and bound with a plastic comb. The manual starts off with details of making a backup copy, followed by a summary of the available commands, and the syntax of any parameters.

The next section details all the *CONFIGURE options, followed by a list of the editor control keys. Then comes the description of the commands. For each command, the syntax is given followed by a plain language description of what the command does, and then an example, a list of possible errors, and any extra notes.

The appendices consist of notes on the single line assembler, an explanation of the possible error messages, and instructions for using the hard disc archiver. Finally, there is a fairly comprehensive index.

One omission from the manual is a list of the SWI calls implemented by *Toolkit Plus*. I discovered by pure chance that there are twenty-five of these, and I am sure that at least some of them could prove useful in your own code. Hopefully, Clares will produce some form of technical note covering these.

The actual packaging of our copy of *Toolkit Plus* was in fact a standard *Toolkit* box with a Toolkit Plus sticker over the name. I suspect that this is only a temporary measure.

CONCLUSION

I was rather disappointed with *Toolkit Plus*. I had been led to believe that the new module would be a great enhancement over the original *Toolkit*. Instead, the changes are largely extensions to the existing features, though the hard disc archiver is a new and useful feature. *Toolkit Plus* costs £49.95 inc. VAT, which, considering current trends in Archimedes software prices, is reasonable. However, users of the existing *Toolkit*, will have to pay £24.95 to upgrade to *Toolkit Plus* - a figure that I find excessive.

If you need a Toolkit package, then *Toolkit Plus* is really the only commercial one available, although you could always use our very own *RISC User Toolbox*. I find the features of *Toolkit Plus* useful, and I would not resent having to pay £50 for it. But if I had already bought *Toolkit*, I would be very reluctant to part with another £25 for an enhanced version which many companies would offer at a nominal cost to registered users.

Product	Toolkit Plus
Supplier	Clares Micro Systems 98 Middlewich Road, Rudheath, Northwich, Cheshire CW9 7DA. Tel. (0606) 48511
Price	£49.95 inc. VAT

RU

MAKING USE OF FILE TYPES

**The Arthur and RISC OS operating systems support the use of file types.
Lee Calcraft explains how they work, and how to use them.**

WHAT ARE FILE TYPES?

Almost all files created by the Archimedes and stored on disc, RAM disc or Econet are given a so-called File Type. This is a number between zero and 4095 (000-FFF hex) which is used to identify the file, and to determine the way in which it will be treated when it is loaded or run. Table 1 gives a summary of the major file types. Acorn have allocated types in the range &000-&7FF to the user and &800-&DFF to software houses, reserving &E00-&FFF for their own use. The only files which cannot be allocated a file type are those containing pure machine code which must be loaded at an address other than &8000. This is because the part of the disc catalogue entry normally used for storing the file type is needed to store the load address of the machine code.

FFF	Plain ASCII Text
FFE	Command (or Exec) file
FFD	Data
FFC	Transient Utility
FFB	Tokenised Basic Program
FFA	Relocatable Module
FF9	Sprite or Saved Screen
FF8	Absolute Code (to run at &8000)
FF7	BBC Font
FF6	Fancy Font
FEC	WIMP Template files
FEB	Obey Files

Table 1. Commonly used file types

If you only use the Archimedes to run proprietary software, then you will probably not come into direct contact with file types. But they are at work nonetheless. It is the file type of a file which tells the Desktop which icon to display when representing it - and this is of much greater importance with RISC OS, where the Desktop becomes central to the operation of the whole machine. The file type is also used to determine what will happen when you *RUN or *LOAD a given file. If file types are correctly set, then when you attempt to execute (i.e. *RUN) a data file, the corresponding application program will be loaded and run, and the data file will then be loaded into it.

Additionally, file types can be used when writing your own applications. You can assign file types in the range 0 to &7FF, and then set up the machine to respond to these files in any way that you wish. Moreover, you can put a simple file type test into a program to ensure that it does not load in the wrong kind of file when loading user-selected data files.

SETTING FILE TYPES

Proprietary applications on the Arc (including Basic itself) automatically assign appropriate file types when files are saved. For example, when you save a Basic program, it will always be given the type &FFB. In some circumstances it is necessary for the programmer to set file types himself. For example, when writing an RMA module it must be specifically designated as type &FFA. And if you try to *RMLoad a file into the RMA which is not of this type, you will get the message:

This is not a relocatable module

There are two principal ways to set the type of a file. The simplest is to use the command:

```
*SetType <filename> xxx
```

where <filename> is the name of any file, and may include a pathname, and xxx is the file type. For example:

```
*SetType $.WORDPROC.MyData 6FF
```

This will set the file in directory $.WORDPROC named MyData to type &6FF.

Alternatively, you can save a file and type it at the same time using the following OS call:

```
SYS "OS_File",10,<filename>,<filetype>,,
<startaddr>,<endaddr>
```

where <filename> is the name under which the file is to be saved, <filetype> is its type, and <startaddr> and <endaddr> are the start and end addresses of the block in RAM where the data resides. For example:

```
SYS"OS_File",10,"MyData",&6FF,,&A000,&A
FFF
```

This will save the contents of RAM between &A000 and &AFFF under the file name MyData, and assign a file type of &6FF to it.

READING FILE TYPES

Unfortunately there is no easy way of reading file types. You must use a SYS call

from Basic (or the SWI equivalent from machine code). To make this easier, I have parcelled up the call into a Basic function called FNreadtype. The accompanying program makes use of this, requesting a filename of the user, and displaying the corresponding file type.

The function is very easily incorporated into other programs. It is called with the file name as a single parameter, and returns its file type as a number between 0 and 4095. Additionally, it sets the variable *filelen* to the length of the target file. As a further example of its use, the following line will check whether the file *MyData* is of type &6FF:

    IF FNreadtype("MyData")=&6FF THEN PRINT
    "ok"

SETTING LOAD AND RUN ACTIONS

On both the Arthur operating system and on RISC OS, the user may determine exactly what happens when a given file is *RUN or *LOADed. This is controlled by the Alias$ system variables. For example, if you enter the following:

    *Set Alias$@RunType_6FF MyApp %*0

This tells the filing system that whenever it receives the instruction:

    *RUN MyFile

or *MyFile

for a file of type &6FF, it must substitute the following:

    *MyApp MyFile

In other words it will try to run the application itself, and load in the data file MyFile. The %*0 means that all parameters given with the original call, including the file name itself, will be passed across to the application program. It is by such means that the programmer can

determine that double-clicking on a given application data file on the RISC OS desktop will start the application itself - since double-clicking a file icon has the effect of *RUNning it.

It is also possible to set Load Types in a similar way. The following sequence will cause any *Load action on data files of type &6FF to run the application MyApp, and load them into it.

    *Set Alias$@LoadType_6FF MyApp %*0

The user may set load and run types for any file type that he wishes. These may even override those set up when the Archimedes initialises. The best place to perform such settings is in a boot file in the root directory of a disc, or in RISC OS !Boot and !Run Obey files in an application directory.

For further details on file types, the reader is referred to the *Programmer's Reference Manual (part 1)*.

```
  10 REM >WotType
  20 REM File Type Reader
  40 :
  50 PRINT'"File Type Reader"
  60 *CAT
  70 REPEAT
  80   INPUT'"Filename",file$
  90   type=FNreadtype(file$)
 100   PRINT"File type= ";
 110   IF type>0 THEN PRINT~type ELSE P
RINT"Machine Code"
 120 UNTIL FALSE
 130 :
 140 DEFFNreadtype(name$)
 150 SYS "OS_File",5,name$ TO,,loadaddr
,,filelen
 160 IF (loadaddr>>>20)=&FFF THEN:=&FFF
AND loadaddr>>>8 ELSE:=-1
```

RU

Points Arising....Points Arising....Points Arising....Points Arising....

ADFS FILE-FIND UPDATE by Lee Calcraft

The short ADFS File-Find utility in Volume 1 Issue 9 fails to work properly with certain nested directory structures. This is remedied by making the four arrays dir(), dir$(), file() and file$() LOCAL to the recursive procedure PROCsearch. To do this, alter line 190 to read:

    LOCAL n,dir(),dir$(),file(),file$()

and move lines 100 and 110 to 194 and 196.

We are grateful to David Pringle for pointing out that, although the program will work on Econet, it will only read the first 32 items in any directory, because of the different way in which OS_GBPB works on the network. The File-Find command in last month's Toolbox does not suffer from this limitation because it uses the call in a different way.

RU

Reviewed by David Spencer

Silicon Vision's *RiscFORTH* is the first and so far only implementation of Forth for the Archimedes. This review will concentrate not only on *RiscFORTH*, but also on Forth itself for those who may be unfamiliar with this language.

FIRST, SECOND, THIRD ...

Forth is quite an interesting programming language, not least because of its total non-similarity to most other languages, except perhaps Prolog. Forth was designed by one Charles Moore in the early seventies, and got its name from the fact that Moore saw it as the first fourth-generation language. However, the IBM mainframe it was developed on allowed only five letter filenames, and therefore the 'u' was missed out.

As mentioned above, Forth is unlike any other modern high-level language, and indeed is also unlike any assembler language. In fact implementations of Forth cannot even by classed as either interpreters or compilers (See page 17 of RISC User Vol.2 No.2 for the difference between these). It is instead a type of language known as a *Threaded Interpretive Language* (TIL). A TIL consists of a set of *Words* linked together in a linked list (hence the term threaded). Each Word consists of a header that contains the name of that Word and a link to the next Word, and a body which contains the machine code instructions to define the task performed by that particular Word. A simple Word, such as one to add two numbers together, would normally just contain the code to perform the operation directly, whereas a more complex Word, such as one to print out a numeric value, might in turn call other Words to perform the task.

The concept of a program doesn't exist as such in a TIL. Instead, a complex task is performed by defining a new Word which makes calls to existing Words in order to perform that task. The process of creating a new Word is called 'compiling', and usually consists of creating a new entry in the linked list of Words, and building up the machine code instructions to call the Words that make up the new Word. Once a new Word is created it can be treated in the same

way as existing Words, and can therefore be used in further definitions of other, more complex, Words.

Forth also differs from other languages because arithmetic expressions are represented using a form called *Postfix Notation*, or more commonly, *Reverse Polish Notation*, invented by the Polish mathematician Lukasiewicz. If you consider the operation of addition in a normal language, the two operands appear on either side of the plus sign, for example:

    A = 12 + 23

However, with Reverse Polish, all operands are stored on a stack. When an operation is performed the appropriate number of values are pulled from the stack, and the result is returned to the stack. In this form, our addition operation would look like:

    12 23 +

This starts off by pushing first the value 12, and then the 23, onto the stack. The add operator then removes these two values from the stack, performs the addition and puts the result back on the stack. Assuming that the stack was empty at the start, then it will contain the single value of 35 when the above operation has been performed. This value could then be pulled off and stored in a variable if desired. This method is extremely efficient to execute.

STANDARDS

The problem with a language such as Forth that can be so easily modified and extended, is that every user has a different version of the language. To overcome problems caused by this, attempts have been made to standardise the Forth language. Unfortunately, there are two differing standards. The official Forth standard, is called Forth-83. The opposing standard is Fig-Forth, named after the *Forth Interest Group*. The latter has lost popularity in recent years.

I shall now concentrate on *RiscForth* itself. This is an implementation of Forth-83, with a number of system dependent extensions to take full advantage of the Archimedes' features. The software is supplied on a single 800K disc, along with the manual, in a white plastic 'video case'. The system is started simply by booting the disc.

EDITORS AND SCREENS

As programs don't exist in Forth, neither can the concept of saving a program. Instead, Word definitions can be stored as plain text in the so-called *Screens file*. This is simply a file containing a variable number of 1K chunks of text, arranged as 16 lines of 64 characters. *RiscFORTH* includes a screen editor (written in Forth) to manipulate the screens files. Once saved, selected screens can be loaded and compiled. Word definitions can also be saved as normal text files.

ASSEMBLER

RiscFORTH also includes an ARM assembler, which in addition to normal instructions also supports the floating point opcodes. The assembler adheres to the doctrine of Reverse Polish, and hence its syntax will seem a bit strange.

STAND-ALONE CODE

The only way to produce stand-alone code from Forth is to save an entire copy of the language. This would normally result in a breach of copyright, however, Silicon Vision have provided a way around this problem. By using a special Forth Word it is possible to prevent the user from accessing directly any of the standard Words of the language. Provided this is done, you are free to distribute a 'copy' of the language.

MULTI-TASKING

RiscFORTH provides a form of multi-tasking. This is a co-operative system, similar in some ways to that offered by RISC OS, although *RiscFORTH* doesn't restrict you to a WIMP environment. Instead, multi-tasking is achieved by defining a set of Words, each being an infinite loop, and each containing a special Word that switches between tasks. Each Word is then executed in turn in a *Round-Robin* type of arrangement. The multi-tasking system is particularly effective for the execution of multiple real-time applications.

RiscFORTH contains a demonstration of multi-tasking in which a Mandelbrot fractal is drawn on the screen, while other Words can be executed simultaneously in a text window at the bottom of the screen.

DOCUMENTATION

Forth is not an easy language to learn, especially if it is your first programming language. Therefore, you might need to resort to a tutorial guide on Forth at an early stage. However, don't expect to find such a book supplied with *RiscFORTH*. Instead, the 163 page A5 format manual is very much a reference guide.

The main section, consisting of about half the manual, is a glossary of all the Words defined in the language as standard. The descriptions of each Word are fairly brief, and I have discovered by delving into the code itself that there are some Words which are not documented at all, although these are mostly low-level system Words. One very useful feature of the manual is that the ASCII character set is printed in order at the foot of each page in this section. Many Forth words start with non-alphabetical characters, and this guide makes it very easy to locate them in the list.

Other sections cover the extensions in *RiscFORTH*, such as multi-tasking, the screen editor, the assembler and support for graphics. There are also sections on system dependent features and details of how particular Words are implemented. On the whole, the manual served well as a reference guide, and it even lists some suggested titles for beginners to read. My only quibble is the binding. This is of the plastic comb form, with the comb being several sizes too small for the number of pages. It gets annoying after a while picking up pages from the floor.

CONCLUSION

RiscFORTH provides a good implementation of the Forth-83 standard, and is written in such a way as to make full use of the Archimedes' speed. Being the first version of Forth available makes it very hard to rate the standard of the package, although I could not find fault with it. The price of £99.95 inc. VAT is comparable with that of other Archimedes languages.

Product	RiscFORTH
Supplier	Silicon Vision Ltd., Signal House, Lyon Road, Harrow Middlesex HA1 2AG. Tel. 01-422 2274 or 01-861 2173
Price	£99.95 inc. VAT

RU

SILICON VISION
SOFTWARE FOR THE ARCHIMEDES & BBC

SolidCAD
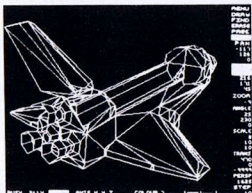
The ultimate 3D Draughting System for Architectural design, Interior design, Engineering Design and Teaching CDT. Allows drawing in plan, front & side elevations and also directly in 3D view. Includes powerful zoom & pan options for precision draughting and surface definition for creating solid colour objects. Also includes Sweep, Extrude & Macro facilities for designing very complex objects easily. Designs created with SolidCAD are compatible with the Realtime Graphics Language for high-speed flicker-free animation. The custom Archimedes version also performs smooth shading for realism. SolidCAD(Arc) users can upgrade to the Realtime Solids Modeller (Arc) for £40.00.

£49.95 (ARC & BBC) New

REALTIME SOLIDS MODELLER

The package includes both the sophisticated design environment of SolidCAD and the high speed animation capability of a Realtime Graphics Language (RGL) module developed in pure ARM Risc code for supercharged performance. The package is ideal for Architectural design, Interior design, Engineering design & teaching CDT. The RGL module can be used to create standalone flicker-free animation of designs from your own programs. Smooth shading is also performed for realistic images. Through our in-house expertise in 3D Design and High-speed techniques, no other package can rival the design environment & animation speed of the Realtime Solids Modeller.

£89.95 (ARC) New

REALTIME GRAPHICS LANGUAGE

The Realtime Graphics Language rom provides a complete 3D Solids/Wireframe animation system with 52 star commands and 3D Editors for designing objects to animate from your own programs. Includes a 35,000 pixels/sec line generator for fast 3D drawing rates, 3D Rotate, Scale, Orbit, Perspective and Turtlegraphics. Also compatible with designs created with SolidCAD (BBC).

£49.95 (BBC)

SUPER-DUMP

The ultimate printer driver which takes advantage of the highest resolution capability of ordinary Epson compatible printers to provide 1920 × 1024 resolution. Images can also be scaled, positioned and previewed before printing. Fully compatible with SolidCAD, Realtime Graphics Language, Gate-Array design system & 3D CAD/Animation system. Your own graphics programs or other CAD packages can be made compatible with Super-Dump by the addition of a few simple commands. An example program is included in the package.

£15.95 (BBC), £24.95 (ARC) New

Presentation Manager

The package provides an interactive environment to create, edit and play-back computer controlled presentations for lectures and demonstrations. Also handles graph plotting for polynomials & user defined functions which can be incorporated within the presentations.
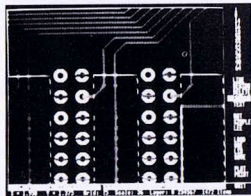
£34.95 (BBC), £49.95 (ARC) New

ARC-PCB

The ultimate PCB design system developed specifically for the Archimedes with a specification that cannot be matched. Includes Automatic routing, Rats-nesting, 8 layers, Surface mount capability, 0.001″ resolution, 32 × 32 maximum board size, On-line Help, Fast Zoom/Pan/Redraw, Text & Silkscreen facility, Variable Line/Pad/Text/Grid sizes, Part Libraries, Block Move/Copy/Rotate/Mirror/Erase options, and up to 300,000 components. For hardcopy, the system supports a large number of plotters and ordinary Epson compatible printers at very high resolutions (1920 × 1024) of 240 dots/inch for near laser quality output. An entry-level ARC-PCB system (without auto-routing) is available for £99.95 (Auto-routing upgrade: £100). Enquire about our turnkey PCB design systems complete with Colour Archimedes and/or plotters with prices from £1220 to £3909 (inc VAT).

£195.00 (ARC) New

RiscBASIC

Supercharge your Archimedes Basic programs by compiling them automatically into pure ARM Risc code with the RiscBASIC compiler. Features include Relocatable modules, Cross reference of all variables, functions, and procedures, Floating point and Integer support, Stand alone code generator, Optimising compiler & Full runtime error handler.

£99.95 (ARC) New

RiscFORTH

A new 32-bit implementation of the FORTH-83 standard, designed to take full advantage of the ARM architecture. Features include Multi-tasking, Optimising compiler, built-in ARM assembler with floating point mnemonics, built-in Full screen Editor, File system interface, OS calls support, Floating point & Integer maths, WIMP support, Single-step debugger, Shadow screen for documentation, Block manipulation, Dictionary & Vocabulary display, Call finding and a standalone code generator.

£99.95 (ARC) New

Postbag

1ST WORD PLUS

Fellow members may be interested in my first impressions of 1st Word Plus. To start with it is incredibly fast when compared with Arcwriter. This latter was definitely tiresome. By half way down the third page I was having to sit back and wait for each character to appear. The longest document I have so far typed in 1st Word Plus has ten pages, and even at the end it was more than keeping up.

The next point is that every command works, unlike Arcwriter where underline, for example, has always been capricious. One slight moan is the prissy way in which 1st Word Plus insists that every document must be saved before it can be printed, but this is only minor.

I have not yet investigated all the facilities, such as mailmerge, because I have no need for that, but on the whole I can really recommend this program.

L.E.Weaver

Arcwriter may not be the strongest competition for 1st Word Plus, but we welcome Mr Weaver's comments. We hope to give greater coverage in future issues of RISC User to commercially available applications software (see article on Logistix in this issue), and would welcome your comments on choice of word processors, spreadsheets, databases and the like for the Archimedes.

FLOATING POINT ARITHMETIC

While ARM Basic is fast by any standards, there comes a time in mathematical problems where more speed is really needed, and I would normally look to assembler to get over the hard bits. Imagine my consternation to learn that the extremely available built-in assembler in Basic V won't touch real numbers.

I was astonished to see that a Mandelbrot program in another magazine employed a binary procedure to allow the use of assembler. Why not use the Floating Point Emulator (FPE) I thought? The explanation came via a friend in the form of a Mandelbrot program which links the FPE with assembler. A nice piece of work yet it only runs twice as fast as my Basic only version.

It seems that the FPE is an old crock put there to keep the seat warm for a co-processor yet to be decided about.

What mathematician would enjoy running his favourite compiled language on an Archimedes at a mere speed gain of 2:1 over Basic? Would it not be possible to dig out the floating point routines which Basic V uses to handle floating point numbers and graft these onto the Basic ARM assembler? Pie in the sky? Well Apple users did it for the 6502, despite total lack of support from Apple itself, and it results in a speed improvement for assembler of five to six times the equivalent Basic program.

Arch Busby

As Mr Busby himself says, the floating point emulator is written to emulate the planned hardware co-processor. This makes the FPE a general purpose maths package conforming to the IEEE standards for floating point representation. The floating point routines in Basic on the other hand are written solely for one purpose. They are written to operate at a lower precision than the FPE, and without all the stringent range checking etc. required of the FPE. Therefore, it is logical that this will prove to be faster than the emulator, although the real hardware co-processor will be many times faster than either. Mr Busby is a little unfair in his speed comparisons. Compiled languages and assembler code are generally several tens of times faster than the Basic equivalent. As for extracting the routines from Basic, this should not prove too hard, and perhaps somebody might like to try it.

MORE UTILITIES

I must qualify as one of the first buyers of the Archimedes (around July/August 1987). I survived a period of no software and inadequate information only because I had my faithful model B. RISC User has, to a large extent, changed that, as it has been very useful in taking owners through a very difficult first year.

My one wish is that you will devote more attention to general purpose utilities for *users*, rather than programmers. In a recent issue, the Disc Menu and Toolbox (disc editor) are the only stand-alone utilities for the non-programmer. However, if someone asks what I do with the Arc, I don't like to tell them that I use it to edit discs.

You have helped us to understand the machine. Now help us to use it.

O.B.Giwa

All reader comment is useful in determining the content of future issues, and this is one request we have already taken note of in our future planning. **RU**

HINTS & TIPS   HINTS & TIPS

Another collection of Archimedes hints and tips rounded up by David Spencer.

LINE IT UP
Matthew White
Because the ARM is a 32-bit processor and RAM is addressed bytewise, it is often necessary to align an address to a word boundary when using data blocks. From within the assembler, the ALIGN directive can be used, but within Basic the easiest way to align an address *addr* to the next word boundary is to use:

```
addr=addr+3 AND NOT 3
```

MORE ALIGNMENT
David Spencer
A cautionary word of warning for assembly language programmers. Basic's assembler will automatically ensure that all instructions are word aligned, so for example:

```
BL grabbyte
EQUB 123
LDMFD R13!,{PC}
```

causes three zeros to be inserted after the value 123 to align the next instruction. However, if the LDM instruction had a label associated with it, then this label *must* appear on the same program line as the instruction. If the label is on a line of its own preceding the instruction then it will be given the value of the program counter *before* alignment. This causes havoc when the machine code is run, and is very hard to spot because it generates no assembly-time errors and is not at all obvious at a first glance. The safest solution is to always put an ALIGN directive after byte and half-word values.

LIST OPTIONS
Glynn Clements
The Basic command LISTO can be used to provide a number of different formats for program listings. The command is followed by a number which specifies the format, the default setting being zero. The number representing the desired format is chosen by adding together the appropriate values from the list below:

1   Print space after line number
2   Indent structures (REPEAT etc.)
4   Split multi-statement lines
8   Don't list line numbers
    (Gives error if any line numbers
    referenced in program)
16 List Basic keywords in lower case

A value of 16 is useful for separating keywords and variable names if you have used upper-case for the latter, while a value of 14 (indent, split and no line numbers)

gives a listing which resembles a Pascal program more than Basic.

RISC OS ASSEMBLER LIMITS
Andrew Johnson
The Basic assembler in RISC OS has been extended to check against the code running over the end of the allocated space. By adding 8 to the OPT setting, the assembler will give an error if the value of P% exceeds that of L%. A statement such as:

```
DIM code% 1000, L% -1
```

will set L% to the address of the end of the area DIMed for code. Therefore, by enabling the checking you can ensure that the code doesn't run over the end of the allocated area.

CLARIFYING COLOURS
Robert Long
Anybody who has tried to manipulate screen memory directly in a 256 colour mode may have discovered that the layout of bits within a pixel is not documented anywhere. The actual allocation for each bit (with the default palette settings) is:

0   Tint low
1   Tint high
2   Red low
3   Blue low
4   Red high
5   Green low
6   Green high
7   Blue high

The two tint bits correspond to the value given to TINT divided by 64. See the descriptions of VDU17 and VDU23 in the *Programmer's Reference Manual* for more details of the 256 colour numbers.

TIME ON A KEY
Graeme Davidson
The commands:

```
*SETMACRO Key$1 <Sys$Date>
*SETMACRO Key$2 <Sys$Time>
```
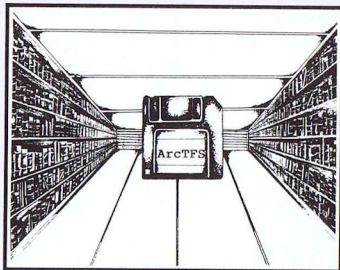
will set up functions keys one and two to produce the current date and time respectively. These could be used to insert the current values into a REM statement to identify different versions of a program under development, or they could be used to insert the current date into a word processor document. **RU**

Dabhand User News

Archimedes Basic Compiler Version 2 • Archimedes Operating System Guide
Special Offers on Archimedes PC Emulator and ANSI C

ABC Version 2

Don't live on promises, buy the only true BASIC V Compiler currently available, and now in its second release! But don't take our word here's what the reviewers said:

"...Excellent Dabs Press product. Buy it!" RISC User Dec.88

"ABC is a vital part of any programmer's toolbox, it puts compilers on other systems to shame. Unquestionably one of the most impressive pieces of software I have yet seen running on the Archimedes." A&B Dec 1988

"...I can tell you now, I am very impressed. This is a superb package." Archive Dec 1988

The above quotes were referring to Version 1 of ABC – ABC Version 2 is even better! Version 2 allows use of double and extended precision floating point, multiple exits from procedures and functions, RETURN parameter passing, new compiler directives and very much more.

As all reviewers have found, ABC makes writing machine code programs and relocatable modules easy. Programs can be written using the full range of BASIC V's error messages and report's, and once fully working, run through ABC which transforms them into machine code, ready for immediate action.

Demo Disc: We have produced a demo disc of ABC which still supports over 100 commands. It costs just £2 and this is refundable on any subsequent purchase. A full specification sheet is also available on request.

Free upgrades!   **£99.95**

Archimedes OS Book

Hot off the presses, this latest Dabhand Guide is essential reading for every user of the Operating System including Arthur *and* RISC OS.

Its 320 pages are packed with vital information and covers topics such as VIDC, MEMC, IOC, Sound, SWIs, Vectors, Filing System and very much more. Price is just **£14.95** or **£21.95** with programs disc and manual.

Archimedes Assembly Language

386 pages devoted to programming the Archimedes in machine code. At **£14.95** it is the *only* book which deals specifically with assembler on the Archimedes. Book and programs disc ordered together – **£21.95.**

Archimedes Games!

Arcendium: Board Game Fun

Backgammon, Draughts, Reversi and Quadline – four games for the price of one! Sure to get the whole family using the Archimedes. **Just £14.95.**

Alerion: Archimedes Arcade Action!

The highly acclaimed all-action shoot-em-up for the Archimedes. 256 colour mode graphics, with digitised speech. Impossible to finish! Superb fun!

"...the gameplay makes this game a winner...a first rate game..." A&B Computing. **Price £14.95.**

C: A Dabhand Guide

PCW said: *"I only wish this book had been available when I was learning C."* It you want to learn ANSI C then this 512 page volume is the way to do it. At **£14.95** it represents quite incredible value. Book and programs disc **£21.95**.

Archimedes PC Emulator Shareware

Five discs full of PC software tested with the emulator to ensure compatibility. The collection includes software you would normally expect to pay a fortune for and includes a wordprocessor, spreadsheet, games, flowchart designer, printer utilities, and more.   **£34.95**

SPECIAL OFFERS! PC Emulator and FREE Shareware

Purchase the Archimedes PC Emulator from us and we'll give you our Shareware pack–described above and worth £34.95 – absolutely free of charge. Price just **£113.85 inc.**

ANSI C and FREE Dabhand Guide to C

The best combination! Purchase Acornsofts ANSI C and we'll throw in our top-selling Dabhand Guide to C – 512 pages and worth £14.95. Price just **£113.85 inc VAT & p&p.**

On the Lookout

Dabs Press are always on the lookout for authors who feel that they have what it takes to write either software or Dabhand Guides. If you have a useful idea then we can make it reality. If you don't have the idea but do have what it takes then we still want to hear from you. Our experience is second to none – if you want to be tops get in touch.

FREE on Request
Bumper Catalogue!

RISC User Magazine Disc
March 1989

IMAGE WRAP AND SPIN A utility to 'wrap' a flat image around the surface of a sphere. The demonstration program uses this to produce images from different angles, giving the effect of a spinning ball.

RISC USER TOOLBOX
The complete Toolbox inc. from this issue a set of disc searching commands and a useful help feature.

MAKING USE OF FILE TYPES
A short program to demonstrate how file types can be read from within your own programs.
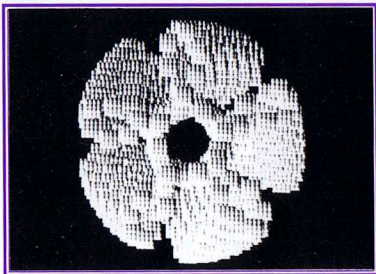
SIMPLE SPRITE MANAGER
A useful module that allows sprite definitions to be displayed before deletion.
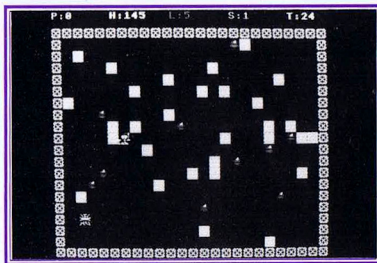
NEW FONT STYLES
A module that provides 72 different styles of the standard screen font.

ARCHIMEDES VISUALS
A sixteen colour fader, and a spirograph which produces an infinite variety of patterns.

✳ BONUS ITEMS ✳

ARC UTILITY
A new version of BEEBUG's public domain file archiver and de-archiver which is fully RISC OS compatible. Documentation is included on the disc.

RAY CHAMBER
A demonstration of the ray tracing capabilities of the Archimedes.

A RISC OS APPLICATION
A sample application, designed to multi-task under RISC OS, which illustrates the use of application directories and the multi-tasking WIMP calls.
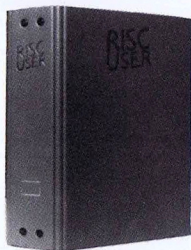
BALLONVANG
An easy-to-play game that is ideal for children.

ARCSCAN DATA
Index entries for this issue of RISC User and the latest BEEBUG (Vol.7 No.8) to be used with *Arcscan*.

RISC User magazine discs are available to order, or by subscription. Full details of prices etc. are given on the back cover of each issue of RISC User.

RISC USER *magazine*

MEMBERSHIP

RISC User is available only on subscription at the rates shown below. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.

RISC USER SUBSCRIPTION RATES

		RISC USER & BEEBUG
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£23.00
£20.00	Rest of Europe & Eire	£33.00
£25.00	Middle East	£40.00
£27.00	Americas & Africa	£44.00
£29.00	Elsewhere	£48.00

BACK ISSUES

We intend to maintain stocks of back issues New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:

Destination	UK, BFPO, Ch.Is	Europe plus Eire	Elsewhere
First Issue	60p	£1	£2
Each subsequent Issue	30p	50p	£1

MAGAZINE DISC

The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.

MAGAZINE DISC PRICES	UK	Overseas
Single issue discs	£ 4.75	£ 4.75
Six months subscription	£25.50	£30.00
Twelve months subscription	£50.00	£56.00

Disc subscriptions include postage, but you should add 60p per disc for individual orders (30p for additional discs on the same order).

All orders, subscriptions and other correspondence should be addressed to:

RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.
Telephone: St Albans (0727) 40303
(24hrs answerphone service for payment by Connect, Access or Visa card)